# Automatic Assignment of Medical Codes

**A Thesis**

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

Mrityunjay Samanta


Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

June, 2021

Supervisors: Dr. Aniruddha Pant, Adwait Bhave

# Certificate

This is to certify that this dissertation entitled **Automatic Assignment of Medical Codes** towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Mrityunjay Samanta at AlgoAnalytics Pvt. Ltd., Pune under the supervision of Dr. Aniruddha Pant, during the academic year 2020-2021.

Dr. Aniruddha Pant

Committee:

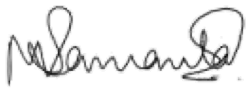
Dr. Aniruddha Pant

Dr. Sourabh Dube

*Dedicated to Ma, Baba and Dada*

# Declaration

I hereby declare that the matter embodied in the report entitled **Automatic Assignment of Medical Codes**, are the results of the work carried out by me at the AlgoAnalytics Pvt. Ltd., Pune under the supervision of Dr. Aniruddha Pant and the same has not been submitted elsewhere for any other degree.



Mrityunjay Samanta

# Acknowledgments

The journey has been long and exciting. This year has brought varied experiences, transitioning from the Physics labs of IISER Pune to the startup culture, from despair to hope, and from rising to flattening curves!!

I want to thank my Mom and Dad for always being there for me. Thank you for absorbing all my tantrums, silently closing the doors during my meetings, bringing me food and water without me having to leave my study table and all the beautiful things you do that I often overlook. I would also like to thank my brother for picking me up whenever I felt frustrated. Thank you for all the lessons you provide on work and beyond. You have been my constant support and offer the much-needed feedback loop.

This work would not have been possible without two great mentors - Aniruddha Pant and Adwait Bhave. Thank you for flattening the steep learning curve of Data Science and making my transition to this exciting and ever-growing field so smooth. The insights, overflowing ideas, suggestions and guidance you provide are invaluable. I want to take this opportunity to thank Prashant, Rahul, Shweta, Pratiti and Aniket from the Algo family for debugging my codes and valuable suggestions. I want to extend my gratitude to AlgoAnalytics for providing me with the opportunity to work and learn in fast-paced startup culture.

Next, I want to thank my various mentors from IISER Pune - Dr Srabanti Chaudhury, Dr Surjeet Singh, Dr Prasenjit Ghosh, Bappa, Divya, and Navita, whom I had the opportunity to work and learn. They have helped me immensely in my academics and research. Additionally, I want to thank Dr Sourabh Dube and Dr Anindya Goswami, who have supported me and given feedback in timely updates whenever I have asked. I would also like to acknowledge various pillars of IISER Pune - Deans, Registrars, Professors, Cleaning and

Canteen Staffs who made IISER Pune my second home. I would also like to thank DST, Govt of India, for providing me with INSPIRE scholarship throughout the BS-MS programme.

The whole work would not have been possible without my Laptop. I have never truly appreciated the things it does for me. Thank you for taking all my frustrations, blows and continuously working even when I am asleep.

I want to thank my constant support system - Aniket, Anoop, Bicky, Gourav, Koustav, Manish, Siddhant, Suraj, Rishabh and Viraj. Thank you for all the support you have provided over the years.

For those of you who I have missed to acknowledge but have played a role in my journey with your encouragement and support, well ...you know who you are, and I am forever grateful to you.
Lastly, my heartfelt tributes to all the COVID warriors - Doctors, Nurses, Scientists, Pathologists, Paramedics, Cleaners and Ambulance drivers.

# Abstract

Natural Language Processing (NLP) is one of the most challenging and rapidly growing fields in artificial intelligence. It is all about deciphering human languages and deriving meaning from them. Some of the commonly used test cases include the classification of sentiments and reviews from text data.

In this study, we present different language models to assign medical codes to electronic health records. Medical codes (ICD codes) are used to map diseases, injuries, health conditions and surgical procedures to a set of universally recognisable alphanumeric codes. They have become essential for storing patient records to analysing health statistics. It also has enormous financial importance in the form of medical billings and insurance. However, assigning codes to medical records are typically done manually and is error-prone due to its complexity.

This work presents a comparative study of machine learning models to assign ICD codes from given medical text with increasing complexity. We believe this research can act as a baseline for further improvements and research.

# Contents

# Chapter 1

# Introduction

Healthcare is rapidly expanding across the world. As more and more people are taking medical facilities, it becomes crucial to maintain the health records of every patient. This will make the healthcare provider more efficient with patient history, will give access to healthcare facilities across regions, serve scientists and governments with epidemiological studies and at the same time help patients with their billing and insurance claims.
Medical coding serves this purpose. In this chapter, we discuss the medical coding methods and challenges faced during the coding process. The chapter also details the dissertation outlook and the outline.

## 1.1 Medical Coding

Medical coding is the process of transforming healthcare diagnosis, procedures, and medical services stored in the digital record (the EHR) of a patient into universal medical alphanumeric codes. An Electronic Health Record (EHR) primarily consists of multiple textual narratives (like discharge summary, medical scan reports, pathology reports, progress notes) authored by healthcare professionals like doctors, nurses, lab technicians, and dieticians.
The content and format of these EHRs depend on the descriptive style of healthcare professional in different regions.
Medical coding happens every time the patient visits the healthcare provider. The healthcare provider reviews the complaint and medical history, makes an expert assessment of what's

wrong, requests tests and scans and documents the entire visit. This documentation helps maintain the patient's ongoing record and maintains all the healthcare services provided by the healthcare provider. This documentation is how the patient is also billed, and the provider is paid.
There are many benefits of maintaing health records and transforming them into small codes. The coding helps in,

- uniformity and sharing of patient data across space and time
- easy storage and access of patient history
- uniform billing, reimbursement of insurance claims
- easy monitoring of the incidence and prevalence of disease history in the region

The medical coding process is carried out by trained and certified medical coders who annotate all patient visits by reviewing EHR manually. These coders rely on their understanding of the procedures and commonality of their specific clinic and facility and designated rule books. When encountered with complex and challenging cases, coders must do in-depth research of a patient's history and check with doctors to arrive at codes correctly. Thus, progress in automated medical coding methods is expected to impact real-world operations for healthcare facilities.

## 1.2 ICD codes

The codes used for EHR coding are International Classification of Diseases (ICD) codes. These codes are regularly revised and maintained by WHO. The latest variant used globally is ICD10. ICD lists different codes for different diseases, injuries and procedures in patients. There are over 70000 codes in ICD10. ICD codes are hierarchical, with top level category being generic referred here as "ICD category" and becomes more specific as one goes down called "ICD codes"(Figure 1.1).

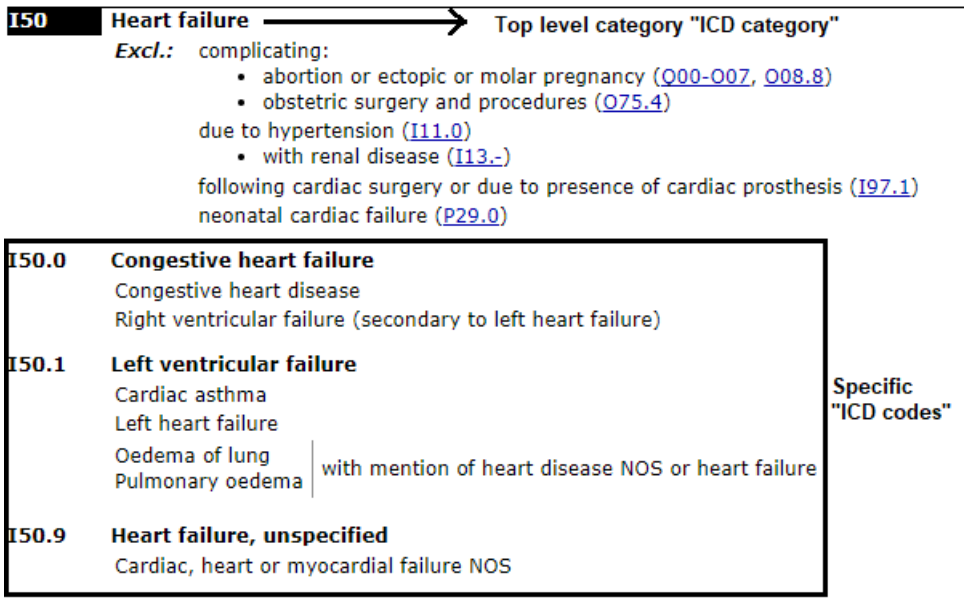



Figure 1.1: Sample ICD10 code with ICD category and specific ICD codes

ICD codes facilitate billing activities, epidemiological studies and also enable researchers to aggregate health statistics and monitor health trends.
To code EHRs effectively, medical coders are expected to have a thorough knowledge of the ICD10 rule book and follow a complex set of guidelines. Several types of error frequently occur when medical coders manually attribute codes:

- ICD codes are hierarchical, top level codes represent generic disease categories and bottom level codes tend to be more specific. Miscoding happens when medical coders assign generic codes to an otherwise specific diagnosis. For example, if a coder accidentally uses the code "heart failure" (ICD10 code I50) instead of "acute systolic (congestive) heart failure" (ICD10 code I50.21). Then the patient may be unable to claim insurance and can be charged substantially more, causing a significant unfair burden.
- Several diagnosis descriptions are closely related to each other. They need to be bundled together rather than finding code for each description. For example, generally, heart disease occurs with hypertension. So if the diagnosis mentions hypertension with heart disease, the medical coder should use the code "hypertensive heart disease" (ICD10 code I11.0) rather than coding "primary hypertension" (ICD10 code I10).
- Similar looking terms may have entirely different codes. For example, if the descrip-

tion mentions “fever with chills”. ICD10 codes can be “fever with chills, unspecified” (R50.9) or can be “typhoid fever” (A01.00) or can be “influenza due to influenza virus” (J11.82). So the coder has to go through the patient’s entire diagnosis to make the claim.

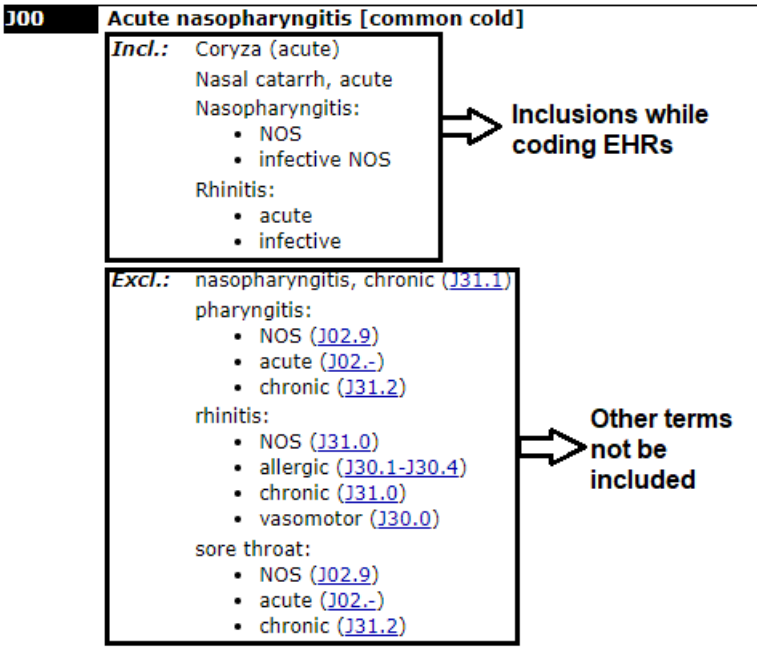



Figure 1.2: A sample ICD code rules with inclusions and exclusions

The coders are required to go through the inclusions and exclusions (Figure 1.2) of each of the 70000 codes and match it with the diagnosis description to assign codes accurately. Once codes are assigned, they are rechecked and reverified for final billing and record-keeping. Therefore, coders and medical providers need to have better tools at their disposal to fasten the process and, at the same time, be accurate.

## 1.3 Thesis Statement

The coding of EHRs is of immense importance for hospital and insurance providers. Machine learning-based methods can be used to create tools to improve the efficiency of medical coders and help reduce coding errors and reduce the unnecessary financial burden on both

the patient and the healthcare provider.
In this dissertation, we try to implement models of different complexities to improve medical coding, with the broad aim of understanding the domain of Natural Language Processing (NLP).
In this dissertation, many significant challenges are encountered when developing automated medical coding methods: handling skewness of ICD codes, medical abbreviations, the relevance of various medical notes. We extract information that we hypothesize are the most relevant in the long medical text and check it on the most frequent labels to overcome the challenges. This is repeated for various models, and at the same time, we try not only to improve the accuracy but also the time efficiency of the models. After all, it should be fast, small, and accurate for AI model to be widely accepted in real-world applications.

## 1.4 Thesis Organization

The chapters in this dissertation are organized as follows:

**Chapter 2** introduces different machine learning terms, methods and algorithms used throughout this dissertation. We then introduce the evaluation metrics used for multi-label classification. We conclude the chapter by discussing the various related work in text classification using the described methods and previous work done in medical coding.

**Chapter 3** begins by introducing the dataset. A detailed data analysis has been performed. The chapter concludes with a description of text preprocessing methods used to extract relevant medical information from medical notes.

**Chapter 4** describes the implementational details and results of various algorithms. The chapter begins with the introduction of NonAI method for the task. This is followed by machine learning algorithms of increasing complexity starting from Logistic Regression to deep learning models like LSTM and GRU and finally to BERT language model.

**Chapter 5** concludes the dissertation by summarizing the results of various models and discussing the scope of improvement and the challenges faced. The chapter ends with a discussion on the future outlook of the study and the state of NLP methods.

# Chapter 2

# Theoretical Background

## 2.1 MultiLabel Classification

MultiLabel classification involves predicting zero or more class labels. In this type of classification, class labels are mutually non-exclusive, which contrasts with regular classification tasks like multiclass and binary classification, where labels are mutually exclusive.


Figure 2.1: Multi Class vs Multi Label (Image Source)

Fig 2.1 shows the difference between MultiLabel and MultiClass. The movie certification

is multi-class as it can be only one of the U, UA, and A, but the movie genre is multilabel as it can have only one genre or range in multiple genres from comedy, thriller, to horror. In our study each medical text would have multiple ICD codes and is discussed in detail in Chapter 3.

## 2.2 OneVsAllClassifier

Not all models support the multi-label classification that we are trying to solve. Algorithms such as Logistic Regression and Support Vector Machines (SVM) are inherently binary classifiers; they do not support more than two classes.

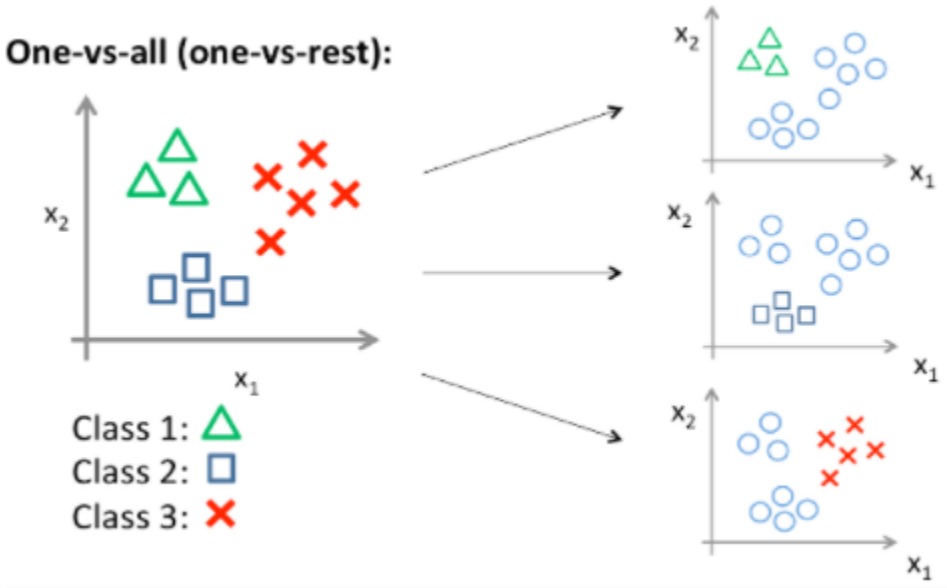



Figure 2.2: One vs All classifier example (Image Source)

One approach for using binary classification for multi-label classification is to split the problem into different binary classifiers, each designed for recognizing one label. This approach is called OnevsAll (one vs Rest). In Fig 2.2, to classify three classes, three binary classifiers are run. Each binary classifier will have one class acting as one label and the other two combined act as another label.
There are many possible downsides to this kind of approach. First, there is that inherent

assumption that labels are not dependent upon one another. Secondly, since this approach requires a classifier for each label, this could be an issue for large datasets and with a large number of labels.

## 2.3 Recurrent Neural Network

Basic neural networks take distinct inputs and try to predict different classes. They assume that inputs are independent of each other. Recurrent Neural Networks(RNNs) are used when inputs are sequential. These are generally used for language modelling because language is a sequence of words and each word is dependent on the previous words. Therefore, RNNs take present input and take input from the last step to process the following input for the next step.

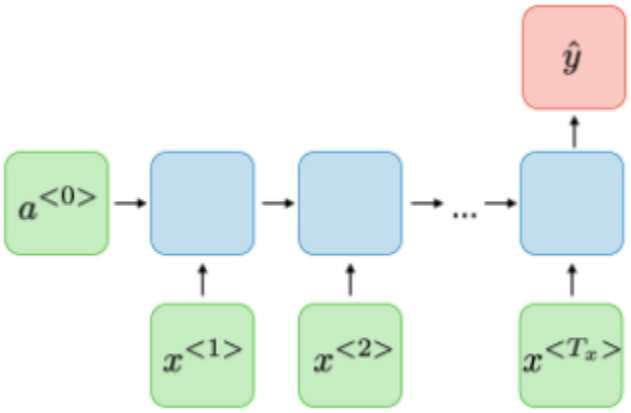



Figure 2.3: Simple RNN architecture (Image Source)

In the above Fig 2.3, $x^{<1>}, x^{<2>}$... represents the input features (these can be sequence of words or any time series data), blue box represents the RNN cell which is a neural network that performs the computation, $a^{<0>}, a^{<1>}$... represents the hidden state of the previous cell. Each RNN cell takes one input feature and one hidden feature (output of the previous layer) to compute the output of the cell which is the input for the next cell. Each hidden state has information of the previous cell and acts as the memory; that is how the RNN network remembers the previous dependencies of the input of sequences. In the last step the network has ideally learnt all the inputs from all the previous layers to give the output.
There are some problems with such models. First, if the sequence is long, the model cannot

learn the long dependencies between the inputs of the sequence. Secondly, not all previous inputs are necessary for a particular input. This is where modified forms of RNN like GRU (Gated Recurrent Unit) and LSTM (Long Short Term Memory) become handy. These modified models are designed to handle long term dependencies and regulate the flow of information only to keep relevant information from previous steps. In these type of networks, there are changes in each cell of the network. These cells have modified formulations called gates, which are a type of multiple layers of neural network. They are designed to control the flow of information from previous cells, how much to pass, how much to forget or to completely ignore.
In all the above models, words need to vectorized. There are various vectorizing methods like one-hot encodings, count vectorizer, TF-IDF and Word2Vec. In the study, we have used TF-IDF and Word2Vec, which are discussed below.

### 2.3.1 TF-IDF

TF-IDF, stands for Term Frequency - Inverse Document Frequency. It is intended to reflect how relevant a term is in a given document.
The intuition behind it is that if a word occurs multiple times in a sentence, we should boost its relevance as it should be more meaningful than other words that appear fewer times (TF). At the same time, if a word occurs many times in a sentence but also in many other sentences in the same document, maybe it is because this word is just a frequent word (like a, an, the); not because it is relevant or meaningful (IDF).

- Term Frequency (tf): gives the frequency of the word in each sentence of the document. It is the ratio of number of times the word appears in a sentence compared to the total number of words in that sentence. It increases as the number of occurrences of that word within the sentence increases.
- Inverse Document Frequency (idf): gives rarity/frequent word in the whole corpus of sentences. It is the ratio of total sentences in the document to the number of the sentences the word appears in.

Example:

A = “the car is driven on the road”
B = “he truck is driven on the highway”

| **Vocab** | **tf_A** | **tf_B** | **idf** | **tfidf A** | **tfidf B** |
|---|---|---|---|---|---|
| car | 1/7 = 0.143 | 0/7 = 0.000 | log2/1 = 0.693 | 0.09902 | 0.0 |
| truck | 0/7 = 0.000 | 1/7 = 0.143 | log2/1 = 0.693 | 0.0 | 0.09902 |
| the | 2/7 = 0.286 | 2/7 = 0.286 | log2/2 = 0.000 | 0.0 | 0.0 |
| on | 1/7 = 0.143 | 1/7 = 0.143 | log2/2 = 0.000 | 0.0 | 0.0 |
| driven | 1/7 = 0.143 | 1/7 = 0.143 | log2/2 = 0.000 | 0.0 | 0.0 |
| highway | 0/7 = 0.000 | 1/7 = 0.143 | log2/1 = 0.693 | 0.0 | 0.09902 |
| is | 1/7 = 0.143 | 1/7 = 0.143 | log2/2 = 0.000 | 0.0 | 0.0 |
| road | 1/7 = 0.143 | 0/7 = 0.000 | log2/1 = 0.693 | 0.09902 | 0.0 |

Table 2.1: tfidf calculation for vectorizing the words.

### 2.3.2 Word2Vec

Word2Vec[1] as the name suggests, is a method for producing vectors corresponding to each word in the corpus. It is a neural network that takes as its input a large corpus of words and produces a vector space, typically of several hundred dimensions. Each unique word in the corpus is assigned a corresponding vector in the space. It is trained to reconstruct the linguistic context of words. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space. Word2Vec comes in two flavours,

- **continuous bag-of-words (CBOW)**: This model takes the surrounding words of each word as the input and tries to predict the word corresponding to the context. So for a given word in the sentence say w(t) (also called centre word or target word), we define a context window C, (say C=5), then the input would be words at positions w(t-2), w(t-1), w(t+1), and w(t+2). CBOW will then try to predict w(t). In the process of predicting the target word, we learn the vector representation of the target word.

- **skip-gram**: This model is just opposite of CBOW. Here instead of taking surrounding words as the input, it takes the target word, w(t) as the input and tries to predict the surrounding words w(t-2), w(t-1), w(t+1), and w(t+2) depending upon the context window (here C=5).

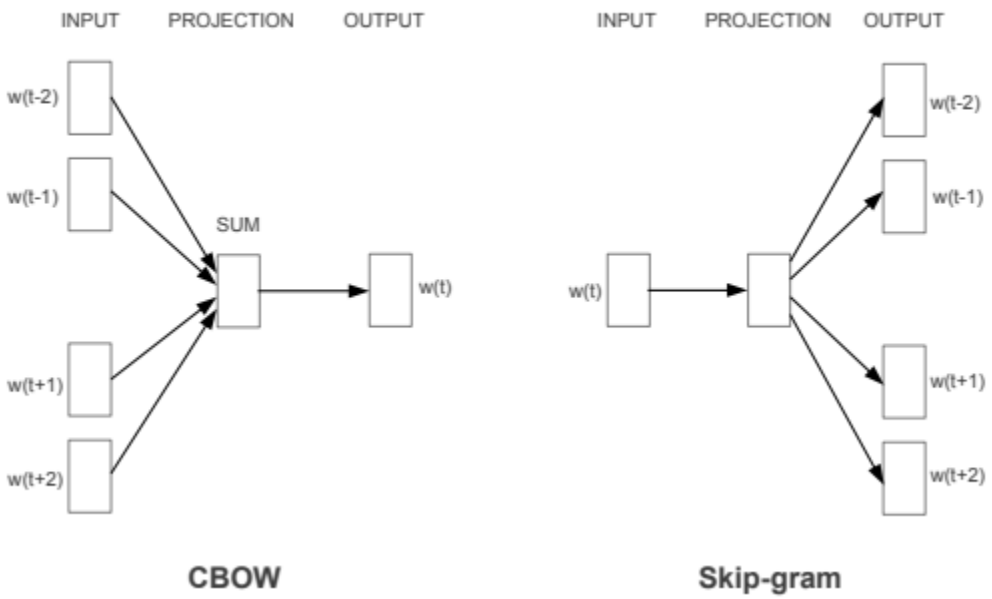



Figure 2.4: Graphical representation of different types of Word2Vec models (Image Source)

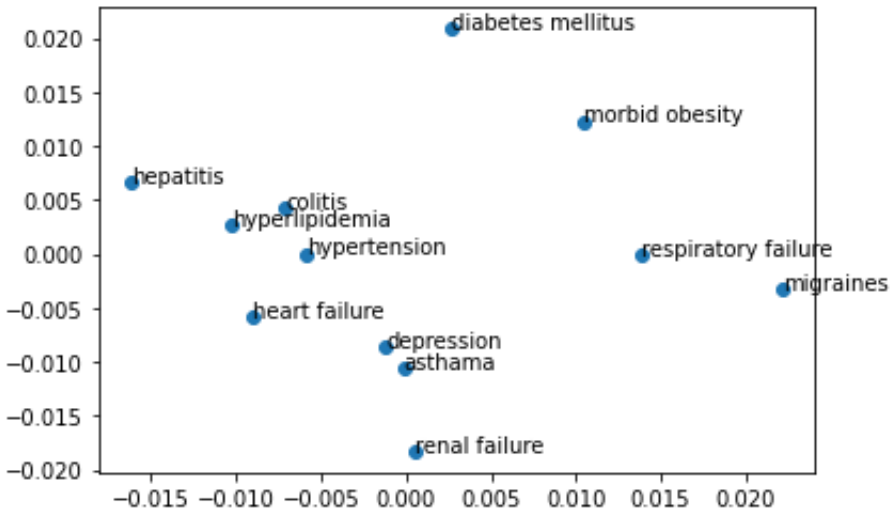



Figure 2.5: word2vec representation in 2-dim trained on medical sentences

These word vector representations can also help in detecting similar words. Such similarity is calculated using cosine similarity given by:
$similarity = \frac{w1.w2}{||w1||.||w2||}$
where w1 and w2 represents the vector representation of words w1 and w2.

## 2.4 Character Level Convolutional Neural Network

Convolutional neural networks(CNN) are generally used in the extraction of information in computer vision problems. They can be extended to text classification using character embeddings [2]. The idea behind character embeddings is to model the lowest representation of words that form the language.
While using word-embeddings discussed in the previous section, a large vocabulary of word is needed for representation of words. This is where character embedding stands out. The size of vocabulary is very small ( 50 for english language including 26 letters, 10 digits, and symbols like ,,/,!,% etc.). Having character embeddings it is possible to generate vectors for every single word even if its out of vocabulary words. It handles misspellings, new words, and exciting/emotional phrases.

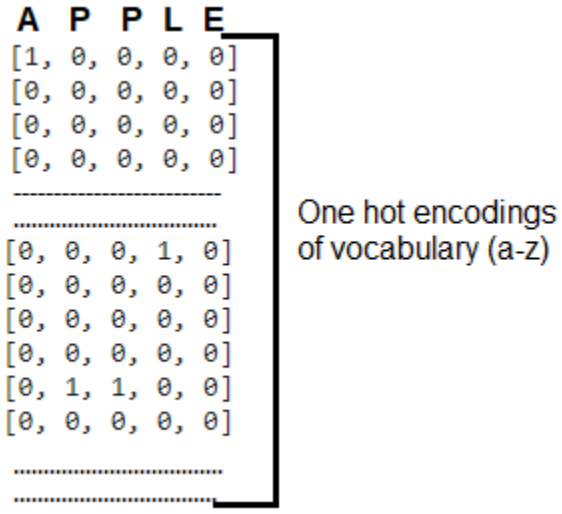



Figure 2.6: OneHot character representation of word "APPLE"

One dimensional CNNs are trained on the character representation of the sentences. They

are capable of extracting local information from input sequence. They are shown to work in the news classfication, sentiment classfication[2].

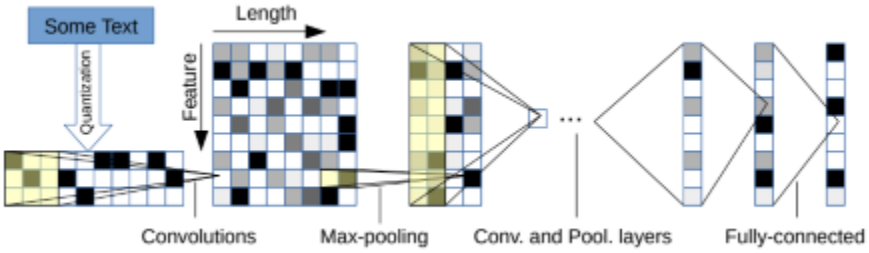



Figure 2.7: CNN architecture on character embeddings (Image Source [2])

The benefits of the character embeddings come at the cost of requiring larger models to fully capture the character level sequences that are slower to train. Nevertheless, these provide a lot of promise with their small vocabulary and flexibility in handling any words, punctuation, and other document structure.

## 2.5 Transformer Models

Transformer models are the latest in language modelling. In most cases, they have improved upon the previous models discussed. Transformer models are based on attention mechanisms. As discussed in the RNN model, language models need to learn from previous words in the sequence; the attention mechanism does precisely that. The attention mechanism does not suffer from short term dependency, unlike RNN. This is still true for GRU and LSTM. As discussed in the previous section, they have a bigger capacity to achieve longer-term memory, therefore having a longer window to reference words, but this window is limited. The attention mechanism, in theory, has an infinite window to reference from. Also, the sequential nature of LSTM and GRU restricts the use of parallelized computational resources. In contrast, the Transformer only performs a small, constant number of steps. In each step, it applies a self-attention mechanism that directly models relationships between all words in a sentence, regardless of their respective position.
BERT (Bidirectional Encoder Representations from Transformers) is based on the principle of transformers. This model developed by GoogleAI has transformed the NLP space. The

challenge in NLP is the lack of enough training data. There are enormous text data available, but domain-specific tasks are left with little data. To help bridge this gap in data, various techniques have been developed for training general-purpose language models using the enormous piles of unannotated text on the web. These general-purpose pre-trained models can be fine-tuned in the smaller domain-specific tasks. BERT is the recent addition to this pre-trained model.
BERT uses a novel technique called masking; it randomly masks words in the sentence and then tries to predict them. Masking means that the model looks in both directions and uses the full context of the sentence, both left and right surroundings, to predict the masked word. This contrasts with LSTM architecture, where context is found out sequentially, not at the same time.

## 2.6 Evaluation Measures

In multi-label classification, a misclassification is no longer a hard wrong or right. A prediction containing a subset of the actual classes should be considered better than a prediction that contains none of them, i.e., predicting two of the three labels correctly is better than predicting no labels at all.

### 2.6.1 Precision, Recall, F1 scores

Precision gives a measure how the model performed with respect to the predictions whereas Recall gives a measure with respect to the actual values. Precision is the number of correctly classified labels to the total number of predicted labels. Recall is the number of correctly classified labels to the actual number of labels. F1 score is the harmonic mean of Precision and Recall.

### 2.6.2 Hamming Loss

Hamming-Loss is the fraction of labels that are incorrectly predicted, i.e., the fraction of the wrong labels to the total number of labels.

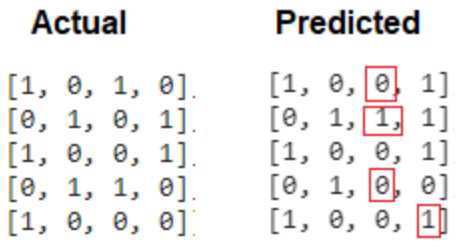



Figure 2.8: Example for showing the evaluation metric

In the above example Figure 2.8:
Number of Actual Positives are 9, Number of predicted positives are 10 and Number of true positives (where actual positives and predicted positives overlap) are 7. $Precision = \frac{Numberoftruepositives}{Numberofpredictedpoitives} = \frac{7}{10}$
$Recall = \frac{Numberoftruepositives}{Numberofactualpoitives} = \frac{7}{9}$
$F1 - score = 2\frac{Precision*Recall}{Precision+Recall} = 0.737$
Number of misclassified labels are 4 (denoted in red).
So the $HammingLoss = \frac{Numberofmisclassifiedlabels}{TotalNumberoflabels} = \frac{4}{20}$

## 2.7 Related Work

The automatic assignment of medical codes has been tried for years. There are essentially two methods used for the assignment of medical codes. One is rule-based methods that use the rules prescribed in the ICD rule book. This requires extreme domain knowledge. These are essentially non-AI methods. Traditionally these methods have been used with the help of the manual intervention of medical coders in real life. They have been able to achieve varied levels of success [3][4][5].
The second method uses learning-based algorithms that are domain agnostic. These learning-based methods range from classical machine learning algorithms to state of the art BERT language model. In classical machine learning models[6][7][8][9], logistic regression, KNNs and SVMs have been tried. They have been found to work on specific manually extracted datasets or a fixed number of smaller labels.
These problems are now being tried using deep learning methods. They have the potential to overcome the barriers of explicit rules and manually extracted features. They have found

to very well generalized across a different kind of texts. A whole range of deep learning methods has been used. Attention-based CNN model[10] have been able to create the state of the art model on a very limited dataset. RNNs, LSTMs and GRUs[11][12][13][14] have also been applied individually and together with other models like attention and pre-trained models. State of the art language model BERT[15] models have so far have given the biggest improvements.

Model results also greatly depend upon the medical text it has been applied to. Different references have applied to different notes of different regions. In our study, we are using the open-source MIMIC III dataset.

Table 2.2 shows the various references that have applied machine learning and deeplearning models to the MIMIC III dataset. It also lists the the section ("discharge summary", "discharge diagnosis") of the dataset that the model was applied to along with range of labels it is trained on. The results shown here are F1-scores.

| **Reference** | **Model** | **Result** |
|---|---|---|
| P Nigam [6](discharge diagnosis, top10 codes) | RNN<br>LSTM<br>GRU | 0.403<br>0.417<br>0.420 |
| Haoran Shi et al.[11], (discharge diagnosis, top50 codes) | Attention Model with LSTM | 0.532 |
| S. Ayyar et al.[16], (discharge summary, top19 categories) | Multiple LSTM stacks | 0.715 |
| Jinmiao Huang et al.[12], (discharge summary, top10 codes) | LR<br>LSTM/Word2Vec<br>CNN/Word2Vec | 0.532<br>0.683<br>0.637 |
| S. Nuthakki et al.[14], (first diagnosis sections, top10 codes) | Transfer Learning with ULMFiT | 0.66 |

Table 2.2: Related Work in Code Assignment in MIMIC III dataset

# Chapter 3

# Dataset

The dataset that is used in the study is the freely available MIMIC III dataset[17]. MIMIC III (Medical Information Mart for Intensive Care III) is a large dataset comprising of de-identified health records of patients at the Beth Israel Deaconess Medical Center between 2001 and 2012. The database contains structured and unstructured data in the form of nursing notes, test results, medications, nutrition, discharge summaries of over 40000 patients and more than 50000 admissions. Each time a patient is admitted to the hospital, there are various notes written by medical staff corresponding to *HADM_ID* (which is the unique admission id) during the stay and is assigned ICD9 codes after discharge.

The whole MIMIC III dataset contains various tables. The medical notes corresponding to each admission are present in the *NOTEEVENTS* table. This table has different categories of text such as Discharge Summaries, Nursing notes, Nutrition plans etc. The different categories of *NOTEEVENTS* dataset with the number of unique patients and admissions is shown in Table 3.2.

The focus of the study is on the *Discharge summary* category of text which is written after carrying out the diagnosis and discharge of the patient. Therefore it contains actual ground truth of the disease/injury of the patient.

In this study, there are two approaches taken to assign ICD codes in the study. The first approach is to treat ICD9 codes independently from each other. The second approach is to group ICD9 codes into top-level categories according to the hierarchy of the ICD codes. The codes generated from the second approach are referred to here as ICD9 categories.

Table 3.1 shows the number of unique patients, admissions, ICD9 codes, and categories in the

MIMIC III. Whole MIMIC III describes the whole dataset, while *NOTEEVENTS* dataset is its subset. Further, *Discharge summary* dataset is the subset of *NOTEEVENTS*.

| **Dataset** | **Patients** | **Hospital Admissions** | **ICD9** | **ICD9 Category** |
|---|---|---|---|---|
| whole MIMIC III | 46520 | 58976 | 6984 | 1070 |
| *NOTEEVENTS* | 46146 | 58361 | 6967 | 1070 |
| *Discharge summary* | 41127 | 52726 | 6918 | 1069 |

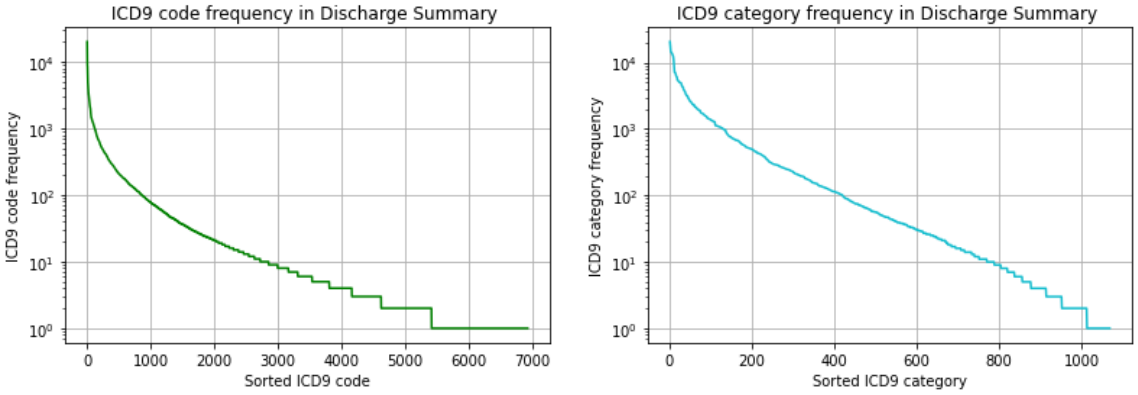
Table 3.1: MIMIC III description

| **Note Category** | **Patients** | **Hospital Admissions** |
|---|---|---|
| Discharge summary | 41127 | 52726 |
| Echo | 22316 | 23585 |
| ECG | 35366 | 44185 |
| Nursing | 7704 | 9070 |
| Physician | 7623 | 8983 |
| Rehab Services | 2130 | 2249 |
| Case Management | 576 | 619 |
| Respiratory | 3598 | 3986 |
| Nutrition | 2823 | 3167 |
| General | 2908 | 3170 |
| Social Work | 1337 | 1393 |
| Pharmacy | 67 | 68 |
| Consult | 49 | 50 |
| Radiology | 37351 | 45526 |
| Nursing/other | 30005 | 34890 |

Table 3.2: Different NOTEEVENTS categories

Understanding the features and distribution of labels is an important step in the making of the model architecture. Here the features are medical text which would be vectorized as discussed in Chapter 2. The labels are the ICD codes and categories. The ICD9 labels in the dataset are very skewed. Few ICD9 codes and categories are very frequent and cover most of

the admissions. Figure 3.1 shows the count of each ICD9 code and category in the *Discharge summary* dataset. The x-axis has each ICD9 code/category number sorted according to the frequency, and the y-axis shows the corresponding frequency on the log scale. The distribution shows a stark difference in the frequency of ICD9 codes and categories for the less frequent code/category.




(a) Distribution of ICD9 codes

(b) Distribution of ICD9 categories

Figure 3.1: Frequency distribution of ICD9 codes/categories

Figure 3.2 describes the fraction of hospital admissions covered with the number of ICD codes and categories. The x-axis has a cumulative number of ICD9 codes/categories, and the y-axis covers the corresponding fraction of admissions covered by it. The top 10 codes and categories together cover more than 70% of the admissions, whereas the top 100 almost cover all the dataset. Table 3.3 is another representation of the above figure that describes the number of admissions covered and corresponding percentages with the top frequent ICD9 codes and categories. Since the top codes and categories together cover most of the dataset and avoid skewness, the study focuses on assigning the top 10 ICD9 codes and categories.

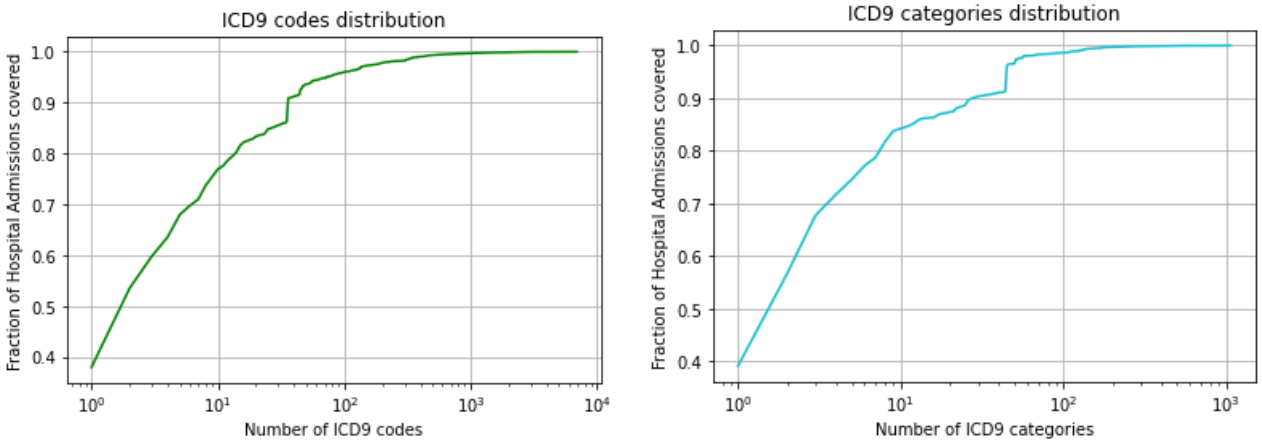
| **Frequecy** | **Admissions** | **%cover** |
|---|---|---|
| Top10 | 40562 | 76.93% |
| Top20 | 43958 | 83.37% |
| Top50 | 49354 | 93.60% |
| Top100 | 50625 | 96.02% |

(a) ICD9 code

| **Frequecy** | **Admissions** | **%cover** |
|---|---|---|
| Top10 | 44419 | 84.24 |
| Top20 | 46089 | 87.41% |
| Top50 | 50903 | 96.54% |
| Top100 | 52007 | 98.64% |

(b) ICD9 category

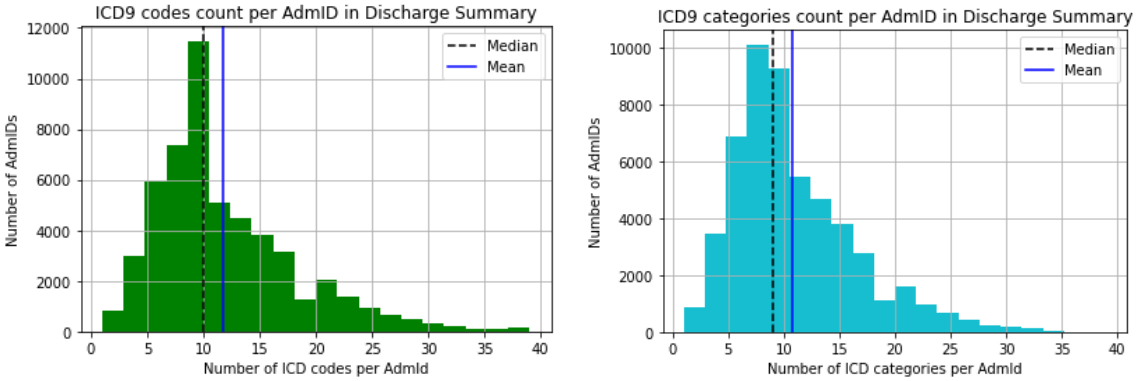
Table 3.3: ICD9 descriptive statistics




(a) Distribution of ICD9 codes (b) Distribution of ICD9 categories

Figure 3.2: Fraction of admissions covered with cumulative number of ICD9 codes/categories

Figure 3.3 describes the number of ICD labels per admission. The graph also presents the mean and median number of labels for each case.




(a) Distribution of ICD9 codes (b) Distribution of ICD9 categories

Figure 3.3: Distribution of ICD9 codes/categories per AdmID

Table 3.4 shows the top 10 frequent ICD9 codes with their definitions. Figure 3.4 shows the disribution of length of *Discharge summary* notes. Here length denotes the entire character length of the text.

| ICD9 Code | Description |
|---|---|
| 4019 | Unspecified essential hypertension |
| 4280 | Congestive heart failure- unspecified |
| 42731 | Atrial fibrillation |
| 41401 | Coronary atherosclerosis of native coronary artery |
| 5849 | Acute kidney failure-unspecified |
| 25000 | Diabetes mellitus without mention of complication-type II or unspecified type- not stated as uncontrolled |
| 2724 | Other and unspecified hyperlipidemia |
| 51881 | Acute respiratory failure |
| 5990 | Urinary tract infection- site not specified |
| 53081 | Esophageal reflux |

Table 3.4: Top 10 ICD9 codes and their description

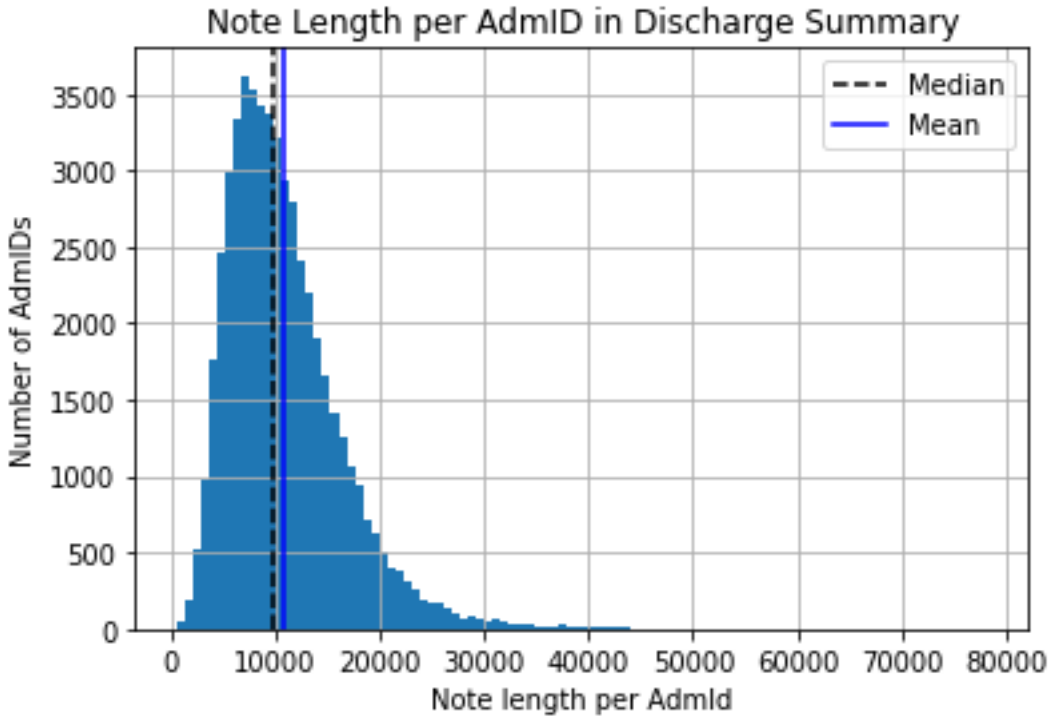



Figure 3.4: Distribution of length of discharge summaries note

## 3.1 ICD10 mapping

It is already discussed in Chapter 1 that the ICD10 is the latest variant. Since the MIMIC III dataset has ICD9 codes, the 100 most frequent ICD9 codes are mapped to ICD10 codes[1]. Table 3.5 shows the sample ICD9 code, its conversion to ICD10 and their corresponding definition. Figure 3.5 describes the distribution of number of ICD10 codes per admission and Table 3.6 shows the admission cover for top frequent ICD10codes.

| ICD9 Code | ICD10 Code | Description |
|---|---|---|
| 4019 | I10 | ESSENTIAL (PRIMARY) HYPERTENSION |
| 4280 | I509 | HEART FAILURE -UNSPECIFIED |
| 42731 | I4891 | UNSPECIFIED ATRIAL FIBRILLATION |
| 41401 | I2510 | ATHEROSCLEROTIC HEART DISEASE OF NATIVE CORONARY ARTERY WITHOUT ANGINA PECTORIS |
| 5849 | N179 | ACUTE KIDNEY FAILURE- UNSPECIFIED |

Table 3.5: Sample ICD9 to ICD10 mapping

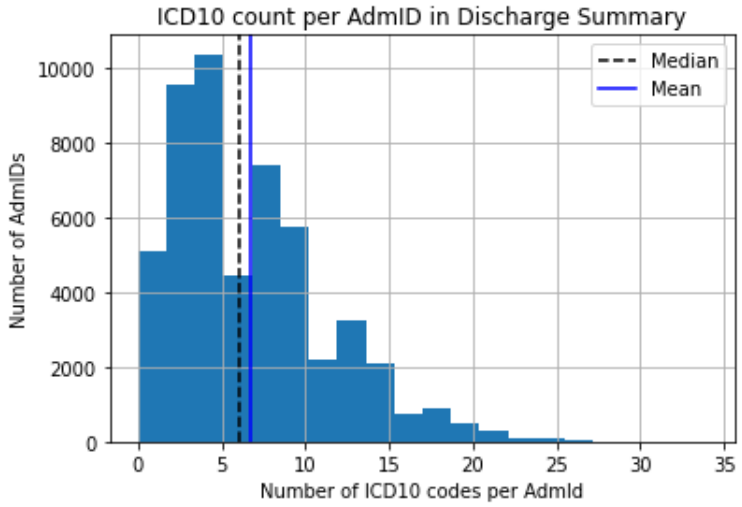



Figure 3.5: Distribution of ICD10 codes per AdmID

[1]ICD9 to ICD10 is not official conversion. The mapping is done through `https://www.icd10data.com/Convert`

| Frequecy | Admissions | %cover |
|---|---|---|
| Top10 | 38954 | 73.89% |
| Top20 | 43764 | 83.0% |
| Top50 | 48199 | 91.41% |
| All ICD10 | 50521 | 95.82% |

Table 3.6: ICD10 descriptive statistics

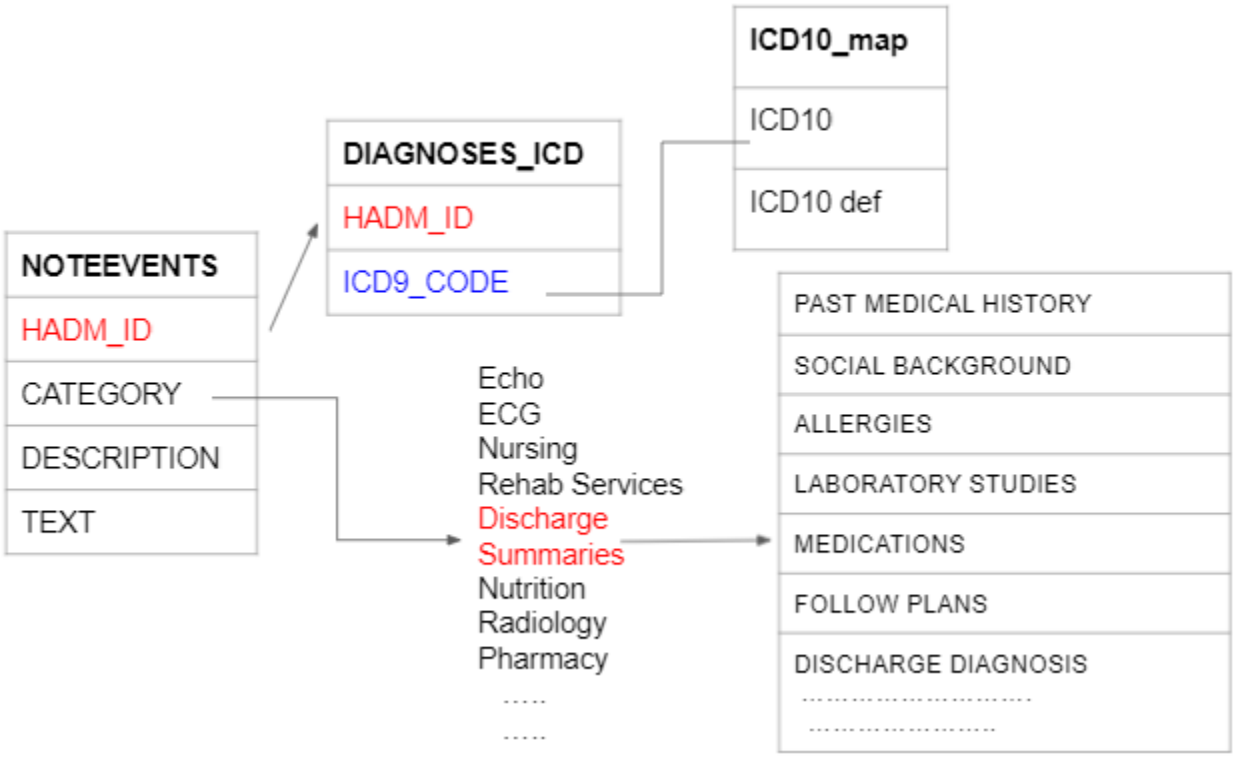



Figure 3.6: The Overall Dataset Schema for the study

## 3.2 Text Preprocessing

Text Preprocessing is the most important step before building a Machine Learning model. The model results can greatly vary on how well the data has been preprocessed. As discussed in text classification problems, features are determined by the vector embeddings, which depend upon the vocabulary of words. Good preprocessing helps reduce the dimensionality of feature vectors and thus help make compact and clean models.

Typical steps in preprocessing are Tokenization, Lower casing, StopWords removal, Stemming and Lemmatization. Tokenization is the splitting of sentences into words. These words help in creating feature space for the model. As the name suggests, lower casing converts word to lower case and helps in bringing uniformity and removing unnecessary duplicates. Words like "Book" and "book" mean the same, but when not converted to the lower case, those two are represented as two different words in the vector space model (resulting in more dimensions). StopWords are the commonly used words (like a, an, the, etc.). They can be extended to include punctuations and numbers, but such removals need to be done with caution. These depend upon the problem being tackled; for example, sentiment classification can benefit from using exclamation (!) and question (?) marks.

Stemming and Lemmatization are text normalizing techniques that convert words into their root form. The difference between them is that Stemming is faster as it cuts the word without taking the context, whereas Lemmatization is slower and expensive as it knows the context of the word before processing.
Lemmatization is more powerful than Stemming. Stemming cuts the suffix of the word either from the beginning or the end. On the contrary, Lemmatization is more involved as it takes into consideration the linguistic analysis of the words. For example, Stemming for "studies" is "studi" whereas Lemmatization returns correct root form "study"; Stemming for "cries" and "cry" returns "cri" whereas Lemmatization returns correct root form "cry".

In the study, our text is the clinical/medical of the *Discharge summary* category from the MIMIC III dataset. Each discharge summary text (Figure 3.7) has various sections like "past medical history", "medications", "discharge condition", "discharge diagnosis". In order to feed to the machine learning models, various sections are extracted from the clinical text. These texts are vectorized using techniques mentioned in Chapter 2. Going through some of the texts, we notice that the text has various abbreviations, common misspellings, important phrases etc.
The various challenges faced in text preprocessing are described below:

- The different sections mentioned above are in no particular order. For example, there is no preference that "Medications" should come before "discharge diagnosis". It is also not mandatory to have all the sections.
- The sections can also have various forms, like text contained in "discharge diagnosis"

is similar to that of “final diagnosis”, “primary diagnosis”, “primary diagnoses” and “secondary diagnosis”.

- There are spelling mistakes like “diagnoses” mispelled as “diagosnes”, “hypercholesterolemia” spelled as “hypercholesterinemia”. There are different short hand writing which depend upon the medical practioner writing the note like many notes have “COPD” whereas somenotes contain full description “Chronic obstructive pulmonary”

- It is not necessary the diagnosis will contain single word diseases like ”hypertension”, it may contain phrases like “ulcerative colitis” , “small cell carcinoma”, “small bowel obstruction”. This should be kept in mind while tokenizing sentences, so that these phrases come together.

The following steps are taken to preprocess the text:

- Lowercasing the whole text, removing the deidentified texts (highlighted in blue in Figure 3.7).

- All the stopwords are removed (like the, they, a, an) and the whole text is lemmatized.

- Common spelling errors are rectified. The errors in medical/scientific terms are hard to figure out and left to the vector representation techniques to make the misspellings similar.

- making use of newline characters to extract different sections and then combining similar ones into one single section. The “discharge diagnosis” contains text from “primary diagnosis”, “diagnosis” etc. Sample extracted “discharge diagnosis” text shown in Figure 3.8.

- removing punctuations like periods and semicolons and numberings from the extracted text.These are futher tokenized to be vectorized using techniques like TF-IDF and Word2Vec.

Date of Birth: [**2080-1-4**] Sex: M

Service: MEDICINE

Allergies:
Patient recorded as having No Known Allergies to Drugs

Attending:[**First Name3 (LF) 1828**]
Chief Complaint:
Mr. [**Known lastname 1829**] was seen at [**Hospital1 18**] after a mechanical fall from a height of 10 feet. CT scan noted unstable fracture of C6-7 & posterior elements.

Family History:
Non contributory

Physical Exam:
Phycial exam prior to surgery was not obtained since patient was intubated and sedated.

Post surgical Physical Exam: (TSICU per surgery team)

Breathing without assistance
NAD
Vitals: T 97.5, HR 61, BP 145/67, RR22, SaO2 98
A-fib, rate controlled
Abd soft non-tender
Anterior/Posterior cervical incisions [**Name (NI) 1830**]
Pt is edemitous in all four extremities, no facial edema
Able to grossly move all four extremities, neurointact to light touch

Discharge Disposition:
Extended Care

Facility:
[**Hospital1 599**] of [**Location (un) 55**]

Discharge Diagnosis:
1. Cervical spondylosis with calcification of posterior longitudinal ligament.
2. Fracture dislocation C6-C7.
3. Ossification of the posterior longitudinal ligament.
4. Aspiration Pneumonia

Figure 3.7: Sample Discharge Summary Text, Sections highlighted in red; De-identified texts in blue

| discharge_diagnosis | ICD9_CODE |
| --- | --- |
| ,1. significant three vessel coronary artery disease status,post coronary artery bypass graft x 4, with left internal,mammary artery to left anterior descending, saphenous vein,graft to diagonal, obtuse marginal and also right posterior,descending artery.,2. hypertension,3. hyperlipidemia,4. 6 cm abdominal aortic aneurysm,5. benign prostatic hypertrophy,6. mild chronic obstructive pulmonary disease,7. 50 pack year smoker,8. history of inferior myocardial infarction and coronary,artery disease,9. hypothyroidism,10. questionable history of osteoarthritis | ['4414', 'V1582', '496', '4019', '2749', '41401', '412', '2449'] |

Figure 3.8: sample extracted "discharge diagnosis" with ICD9 code

# Chapter 4

# Machine Learning Models

In this chapter, details of various models that have been implemented are discussed, along with their results. The study has been done on *"discharge diagnosis"* section of *"Discharge summary"* medical texts. In this study, we solve the multi-label classification for ten frequent ICD codes and categories. As discussed in the previous chapters, each admission in the dataset has multiple labels attached to it. There are essentially two main methods for tackling this kind of classification task. One is transformation into multiple binary classifiers (OnevsAll classifier), and the other making the model learn multi-label classifications using only one classifier.

The chapter starts with the NonAI method that tries to assign codes using the defined ICD rules. This forms the baseline model upon which machine learning models are built. The machine learning models start with Logistic Regression using OnevsAll classifier and proceed to language models of increasing complexity.

For all the models discussed below, the data is split into training, validation and testing in the ratio of 5:2:3. The results are reported on the test set.

All the computations are performed in Google Colab platform. The GPU support whenever needed are also done in Google Colab. All the codes are present in this GitHub Repository.

## 4.1 NonAI

In the NonAI model we try to predict the ICD codes/categories from their frequently associated medical terms. The algorithm steps are discussed below:

1. For each of the code/category we note down the the frequently occuring medical keywords and list them according to the frequency.
2. Then we drop those terms that are common in 80% of the categories and repeat the above until less than the threshold common terms remain for each code/category.
3. Give weightage to the terms according to the frequency of the keywords.
4. Then for each new *HADM_ID* in the test set, calculate the total weights for each code/category based on the medical terms. Then list the ICD codes/categories with highest weightage.

An example of the algorithmic steps of the above NonAI method is shown in Figure 4.1.

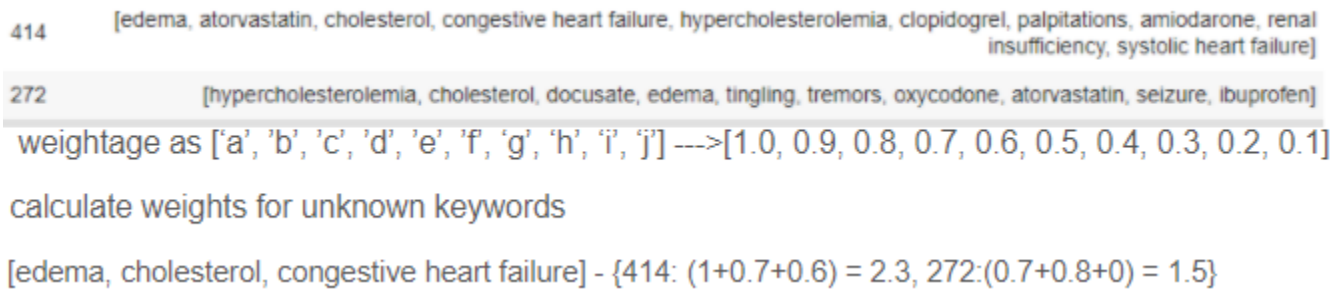



Figure 4.1: NonAI example of calculating weights

The evaluation metrics using this method on test set are shown in Table 4.1.

| **ICD type** | **Hamming Loss** | **Precision** | **Recall** | **F1 - score** |
|---|---|---|---|---|
| ICD9 Category | 0.318 | 0.436 | 0.194 | 0.269 |
| ICD9 Codes | 0.265 | 0.311 | 0.184 | 0.231 |
| ICD10 Category | 0.285 | 0.369 | 0.192 | 0.253 |
| ICD10 Codes | 0.271 | 0.328 | 0.181 | 0.234 |

Table 4.1: Results of NonAI method

The results show that precision is higher than recall for all of the cases. The model has become very picky and predicting very less codes due to the restriction of medical terms placed on it. Whatever it is predicting is almost right, but it also misses a lot of actual codes because it is picky.
To improve upon the above method, similar terms are grouped together. For example "hypertension" generally comes in the patients who already have "diabetes" and "heart failures"; "hepatitis" and "renal failure" are also very similar. This grouping of medical terms are done through pretrained Word2Vec model and calculating the similarity between word vectors using cosine similarity discussed in Chapter 2.

| **ICD type** | **Hamming Loss** | **Precision** | **Recall** | **F1 - score** |
|---|---|---|---|---|
| ICD9 Category | 0.626 | 0.307 | 0.857 | 0.452 |
| ICD9 Codes | 0.685 | 0.222 | 0.863 | 0.353 |
| ICD10 Category | 0.663 | 0.255 | 0.857 | 0.393 |
| ICD10 Codes | 0.678 | 0.233 | 0.859 | 0.366 |

Table 4.2: Results of NonAI method using similarity of words

The results of these are shown in Table 4.2. Here one can observe that recall is way higher than precision. This is exactly the opposite of the first case. This is because it has predicted many labels. It casts a wide net to catch many correct labels but also many other things. The labels corresponding to similar terms are also being predicted. For example, ICD codes related to both heart disease and hypertension come together; therefore, the recall would be higher as it will match one of the codes, but at the same time, false positives will lead to lower precision. This misclassification leads to higher hamming loss.

## 4.2 OneVsAllClassifier

This the first of the machine learning models. In this model we train Logistic Regression classifier for each of the 10 labels. The text features are vectorized using TF-IDF discussed in Chapter 2. The regularization and hyper-tuning of parameters are also done to avoid overfitting. The results are shown in Table 4.3.

| ICD type | Hamming Loss | Precision | Recall | F1 - score |
|---|---|---|---|---|
| ICD9 Category | 0.196 | 0.767 | 0.432 | 0.553 |
| ICD9 Codes | 0.148 | 0.762 | 0.382 | 0.508 |
| ICD10 Category | 0.167 | 0.764 | 0.404 | 0.528 |
| ICD10 Codes | 0.158 | 0.752 | 0.374 | 0.499 |

Table 4.3: Results of OneVsAll classfier

The classifier has become very choosy. This is evident from higher precision and low recall. It casts a very small net of specialised labels, which are correct but also misses a lot of labels. Another drawback of the OnevsAll classifier is that it assumes there's no relationship between the labels. In reality, some labels are together more often than not. As discussed in the above section, ICD codes corresponding to "hypertension" and "heart disease" occur together. The model fails to learn these features.

## 4.3 Character level CNN

The model uses character embeddings for vectorizing the text and is passed to series of one dimensional Convolutional Layers. The Character space used in our model is :

abcdefghijklmnopqrstuvwxyz0123456789
,;.!?:'"/\—_@#$%ˆ&*˜+-=<>()[]{}.

The total vocabulary is 69; 68 of character space and one for denoting blanks and unknowns. As discussed in chapter 2, these type of embeddings can easily denote misspellings and out of vocabulary of words. The words and text are passed as one hot encodings of the character space to the model.
The initial architecture for the model was inspired from the *Yann LeCun and cowrkers* [2] paper. The architecture consisted of 6 Convolutional layers. The model did not work as it overfit our data. Then we tried to train our model on different number of layers from 2 layers to 6 layers with different sizes for each layer. A sigmoid activation function is also applied on the last layer to generate the multi-label output. Among our model architecture,

the best performed model pipeline is shown in Table 4.4. The results of the model on test set is shown in Table 4.5.

| **Architecture** | **Parameters** |
|---|---|
| Conv1d | OutChannels-256, KernelSize-7 |
| MaxPool1d | KernelSize-3 |
| Conv1d | OutChannels-256, KernelSize-7 |
| MaxPool1d | KernelSize-3 |
| Conv1d | OutChannels-256, KernelSize-3 |
| Conv1d | OutChannels-256, KernelSize-3 |
| Linear | Output-1024 |
| Linear | Output-1024 |
| Linear | Output-10 |

Table 4.4: Configuration Details of the Best Performed CNN Architecture

| **ICD type** | **Hamming Loss** | **Precision** | **Recall** | **F1 - score** |
|---|---|---|---|---|
| ICD9 Category | 0.256 | 0.561 | 0.394 | 0.463 |
| ICD9 Codes | 0.178 | 0.615 | 0.308 | 0.410 |
| ICD10 Category | 0.213 | 0.579 | 0.312 | 0.405 |
| ICD10 Codes | 0.187 | 0.613 | 0.234 | 0.397 |

Table 4.5: Results of CNN method

The results of CNN have not been able to improve upon the linear OneVsAll classifier. This shows either there is long way to go to tune the large number of parameters in the model or the character embedding method is not the right vector representation for the medical terms in the medical text.

## 4.4 Word Embedding RNN/LSTM/GRU

RNN/LSTM/GRU methods are the mainstays of text classification. As discussed in Chapter 2, they take sequential data into the model, and that way, they remember the interdependencies of word with respect to other words in the document. We have created two different

models with RNN, GRU and LSTM architectures with different layers and sizes. We have used two different techniques for text vectorization. First, the vectors are initialized randomly and second, we have used pre-trained Word2Vec embeddings. The results on the first kind of embeddings yielded very poor results for all the different types of architectures. For the Word2Vec model, we trained the model on the text of MIMIC III *"Discharge summary"*and passed it onto different model architectures. RNN model still performed poorly, but the results of the LSTM and GRU models improved significantly. Different layer configurations with different hidden sizes have been tried for the LSTM and GRU architectures. Best performing architectures for LSTM and GRU are shown in Tables 4.6 and 4.7 respectively.

| **Architecture** | **Parameters** |
|---|---|
| LSTM | hiddensize - 256 |
| LSTM | hiddensize - 256 |
| Linear | output - 128 |
| Linear | output - 10 |

Table 4.6: Configuration Details of the Best Performed LSTM architecture

| **Architecture** | **Parameters** |
|---|---|
| GRU | hiddensize - 256 |
| GRU | hiddensize - 256 |
| Linear | output - 10 |

Table 4.7: Configuration Details of the Best Performed GRU architecture

The results of the LSTM and GRU models have been shown in Table 4.8 and Table 4.9. The results show a slight improvement from the OneVsAll classifier. There is huge scope in the improvement of results from more fine-tuning parameters like the number of linear layers and hidden sizes. Some overfitting can also be avoided by playing with regularization methods and dropout layers. Nevertheless, the results show that the model learns from the sequential features of the medical keywords. Comparing with the results of the OneVsAll classifier (Table 4.3), one can infer that though sequential modelling of keywords is important, a model representing the individual medical keywords is equally important.

| ICD type | Hamming Loss | Precision | Recall | F1 - score |
|---|---|---|---|---|
| ICD9 Category | 0.208 | 0.717 | 0.424 | 0.533 |
| ICD9 Codes | 0.157 | 0.720 | 0.361 | 0.481 |
| ICD10 Category | 0.178 | 0.703 | 0.407 | 0.516 |
| ICD10 Codes | 0.166 | 0.688 | 0.386 | 0.495 |

Table 4.8: Results of LSTM method

| ICD type | Hamming Loss | Precision | Recall | F1 - score |
|---|---|---|---|---|
| ICD9 Category | 0.199 | 0.751 | 0.429 | 0.546 |
| ICD9 Codes | 0.149 | 0.720 | 0.421 | 0.531 |
| ICD10 Category | 0.170 | 0.741 | 0.414 | 0.531 |
| ICD10 Codes | 0.158 | 0.733 | 0.393 | 0.512 |

Table 4.9: Results of GRU method

## 4.5 Hybrid Model and Transformer Model

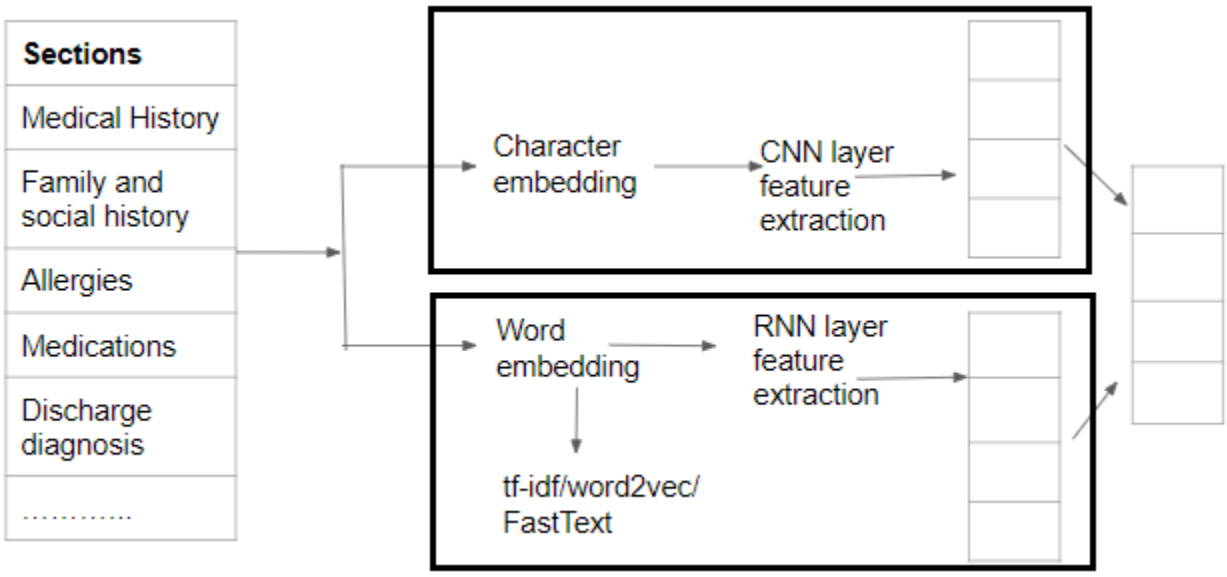



Figure 4.2: Hybrid Model idea of combining CNN and LSTM

We have tried to improve upon the previous models by using the hybrid model and pre-trained BERT model. The idea behind combining LSTM and CNN model (Figure 4.2) stems from the fact the CNN is good at capturing local features (like they can capture lines and edges in images), and LSTM good at classification tasks by learning the context of the words. These type of convolutional-LSTM networks have been shown to improve on text classification tasks like sentiment classfication[18] and text document classfication[19][20]. The results for our model are shown in Figure 4.10 show very marginal improvement upon the LSTM model. As expected, the model is also very memory and time-consuming.

| **ICD type** | **Hamming Loss** | **Precision** | **Recall** | **F1 - score** |
|---|---|---|---|---|
| ICD9 Category | 0.217 | 0.652 | 0.486 | 0.557 |
| ICD9 Codes | 0.158 | 0.659 | 0.445 | 0.531 |
| ICD10 Category | 0.179 | 0.684 | 0.423 | 0.523 |
| ICD10 Codes | 0.161 | 0.712 | 0.392 | 0.505 |

Table 4.10: Results of CNN+LSTM method

The state of art models using transformers are racing ahead in the text modelling tasks. They are pre-trained on millions of parameters and then trained on domain-specific tasks. They have also been used in assigning ICD codes from medical documents [15].
The classification model is pre-trained using the "bert-base-uncased" model [21] which is the most miniature BERT model. The text features tokenized using the same model. The model has also been fine-tuned on the pre-trained parameters. The improvement in results (Fig 4.11) come at the cost of high memory consumption and long training periods.

| **ICD type** | **Hamming Loss** | **Precision** | **Recall** | **F1 - score** |
|---|---|---|---|---|
| ICD9 Category | 0.176 | 0.702 | 0.524 | 0.600 |
| ICD9 Codes | 0.149 | 0.691 | 0.472 | 0.561 |
| ICD10 Category | 0.165 | 0.718 | 0.481 | 0.576 |
| ICD10 Codes | 0.153 | 0.725 | 0.442 | 0.549 |

Table 4.11: Results of BERT method

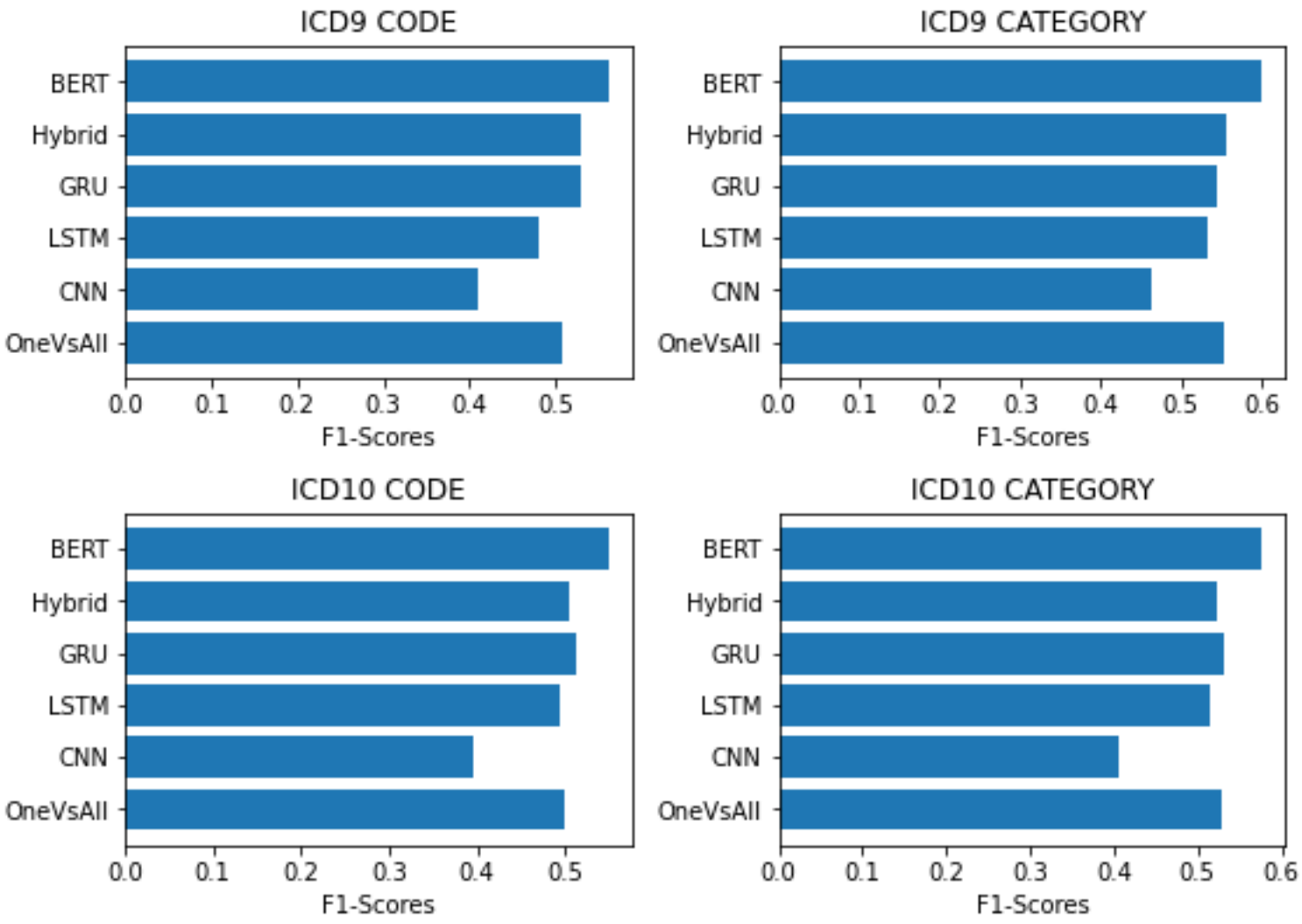



Figure 4.3: Comparing all the models on different ICD types

# Chapter 5

# Conclusion and Future outlook

The study evaluates different machine learning and deep learning models for assigning medical codes to clinical text. The pre-trained BERT model performed the best with F1-score of 0.6 on ICD9 CATEGORY but is limited by the computational and time cost. The GRU, LSTM and hybrid model fared better than the baseline OnevsAll classifier in some cases, but there was not much benefit in capturing long term dependencies as we had hoped for. There is a very high scope of improvements in deep learning models (GRU, LSTM, CNN) with many parameters still left to be played with. OnevsAll classifier performed the best if considered computation and time costs. The classifier has the least amount of hyperparameters to be tuned compared to other models. The classifier is limited by its picky nature as discussed in the previous chapter, restricting the model usage if the text becomes larger.

There is a considerable scope of improvements from the data extraction process to the fine-tuning of the models. The amount and quality of text fed into the model have large implications for the results. The study focusses on the "*discharge diagnosis*" section of the dataset. There are many different medical sections (like past medical history, medications, family history) in the text. These can be permuted to observe what sections are medically relevant for ICD coding and evaluate the results.
Data preprocessing becomes of utmost importance in these language modelling tasks. One of the significant limitations of our dataset is it is very domain-specific. Transfer learning models also perform poorly in such cases. Medical data are uniquely placed. Their format depends on the medical practitioner writing it. As a result, they have a very poor general-

ization. While tokenizing such sentences to words for vectorization, phrases are important, and special care needs to be taken care to avoid splitting of phrases. The phrase "diabetes mellitus type ii" has a special meaning when those words occur together rather than "diabetes" itself; likewise, the phrase like "right medial frontal lobe" and many more.
Furthermore, there are many abbreviations and shorthand writings in the medical document. Developing a better vocabulary that would include these facets, perhaps by some specialist medical practitioners, will help. These will create better embeddings and features for the model, leading to a better deep learning model and results.

We also need to accurately learn essential words from the rules that make up ICD codes to use them in the text features of the model. A more complex architecture in LSTM and RNN layers with finely controlled features from these rulebooks will certainly help. This is where pre-trained transformer models come to the fore. Models like BERT, ELMo, GPT have been trained on billions of text features, but they are constrained computationally.

We believe the deep learning models need to be short and sweet. At a larger scale, the NLP community has been trying to make the size of the models smaller. The medical domain has an immense future for deep learning tasks. These applications include analysis of medications with test results and vital signs, identifying correct people for clinical trials by looking at their medical history and understanding patients at greater risk of catching new diseases by looking at their health records. We believe this study will further help the NLP research community to make better models to improve upon one of the most vital and growing parts of human lives.

# Bibliography


[1] Tomás Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[2] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626, 2015.

[3] L. S. Larkey and W. Croft. Automatic assignment of icd9 codes to discharge summaries. 1995.

[4] Berthier Ribeiro-Neto, Alberto H.F. Laender, and Luciano R.S. de Lima. An experimental study in automatically categorizing medical documents. *Journal of the American Society for Information Science and Technology*, 52(5):391–401, 2001.

[5] Julia Medori and Cédrick Fairon. Machine learning and features selection for semi-automatic ICD-9-CM encoding. pages 84–89, 06 2010.

[6] P. Nigam. Applying deep learning to icd-9 multi-label classification from medical records. 2016.

[7] A. Perotte, R. Pivovarov, K. Natarajan, N. Weiskopf, F. Wood, and N. Elhadad. Diagnosis code assignment: models and evaluation metrics. *J Am Med Inform Assoc*, 21(2):231–237, 2014.

[8] Yitao Zhang. A hierarchical approach to encoding medical concepts for clinical notes. In *Proceedings of the ACL-08: HLT Student Research Workshop*, pages 67–72, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[9] Alan R. Aronson, Olivier Bodenreider, Dina Demner-Fushman, Kin Wah Fung, Vivian K. Lee, James G. Mork, Aurélie Névéol, Lee Peters, and Willie J. Rogers. From indexing the biomedical literature to coding clinical text: experience with MTI and machine learning approaches. In *Biological, translational, and clinical language processing*, pages 105–112, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[10] L. Cao, D. Gu, Y. Ni, and G. Xie. Automatic ICD Code Assignment based on ICD's Hierarchy Structure for Chinese Electronic Medical Records. *AMIA Jt Summits Transl Sci Proc*, 2019:417–424, 2019.

[11] Haoran Shi, Pengtao Xie, Zhiting Hu, Ming Zhang, and Eric P. Xing. Towards automated ICD coding using deep learning. *CoRR*, abs/1711.04075, 2017.

[12] Jinmiao Huang, Cesar Osorio, and Luke Wicent Sy. An empirical evaluation of deep learning for ICD-9 code assignment using MIMIC-III clinical notes. *CoRR*, abs/1802.02311, 2018.

[13] Anthony Rios. Deep neural networks for multi-label text classification: Application to coding electronic medical records. 2018.

[14] Siddhartha Nuthakki, Sunil Neela, Judy W. Gichoya, and Saptarshi Purkayastha. Natural language processing of MIMIC-III clinical notes for identifying diagnosis and procedures with neural networks. *CoRR*, abs/1912.12397, 2019.

[15] Saadullah Amin, Gunter Neumann, Katherine Dunfield, Anna Vechkaeva, Kathryn Annette Chapman, and Morgan Kelly Wixted. Multi-label classification of ICD-10 codes with BERT.

[16] Sandeep Ayyar and Oliver Bear Don't Walk IV. Tagging patient notes with ICD-9 codes.

[17] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, 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.

[18] Mehmet Umut Salur and Ilhan Aydin. A novel hybrid deep learning model for sentiment classification. *IEEE Access*, 8:58080–58093, 2020.

[19] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. A c-lstm neural network for text classification, 2015.

[20] Guibin Chen, Deheng Ye, Zhenchang Xing, Jieshan Chen, and Erik Cambria. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2377–2383, 2017.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.