Medical code classification based on free-text clinical notes

Author

Guillaume KUNSCH

Supervised by

Dr Alexander BELIKOV

A Thesis submitted in fulfillment of requirements for the degree of
Master 2 Artificial Intelligence, System, Data

Department of Computer Science
PSL University
2023

Abstract

The automatic classification of medical codes has recently been an active area of research for the NLP community. The most recent methods have treated this problem as a multi-class multi-label classification problem and have proposed various architectures to solve it. However, these architectures have either not exploited the superior performance of pre-trained language models (RoBERTa) or have not made use of the medical code classification hierarchy.

Consequently, this work aims to reconcile the two approaches by, on the one hand, extracting relevant textual features that overcome the classical obstacles of automated medical coding (large label space, long input sequences and domain mismatch between pre-training and fine-tuning) and, on the other hand, exploiting classifications hierarchy through Poincaré embeddings in hyperbolic space. In particular, we propose a new hierarchical attention decoder that outperforms the current SOTA, and we highlight the difficulties encountered when working with non-Euclidean geometry.

Furthermore, unlike most scientific publications, we will focus not only on ICD codes but also on CPT codes. In addition, thanks to Qantev's partners, we show that mixing MIMIC benchmark data with proprietary data leads to better results.

Contents

1 Introduction

Contents

1.1 Medical coding

Medical coding involves transforming medical diagnoses, procedures, services and equipment into universal medical alphanumeric codes. Diagnoses and procedure codes are derived from documents in the medical record, such as doctor's note transcriptions, laboratory and radiology results, etc. Figure 1.1 presents a case using only text as input but images (CT or MRI scans) could be used as well. Medical coding professionals ensure that codes are applied correctly during the medical billing process, which includes extracting information from documentation, assigning the appropriate codes and creating a claim for reimbursement by insurance companies.

Medical coding has its origins in the public mortality records posted in London in the 19th century. It was by studying them that John Snow determined the cause of a cholera epidemic [1]. Medical coding is even more vital today, as the data collected through it is used to improve overall healthcare. Results are submitted to payers for reimbursement, but code-derived data is also used to determine utilization, manage risk, identify resource use, build actuarial tables and support public health action.

Several classification systems coexist, such as APC (Ambulatory Payment Categories), CDT (Code on Dental Procedures and Nomenclature) and MS-DRG (Medical Severity Diagnosis Related Groups). For our purposes, we will focus on the two most widely used classifications in the world: ICD and CPT codes.

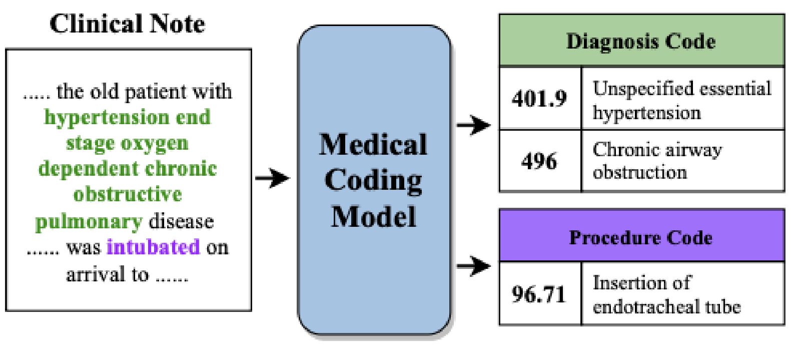

Clinical Note
..... the old patient with hypertension end stage oxygen dependent chronic obstructive pulmonary disease ...... was intubated on arrival to ......
Medical Coding Model
Diagnosis Code
401.9
Unspecified essential hypertension
496
Chronic airway obstruction
Procedure Code
96.71
Insertion of endotracheal tube

Figure 1.1: A medical coding model maps an example clinical note to the corresponding ICD procedure and diagnosis codes. Source:[2]

1.1.1 ICD codes

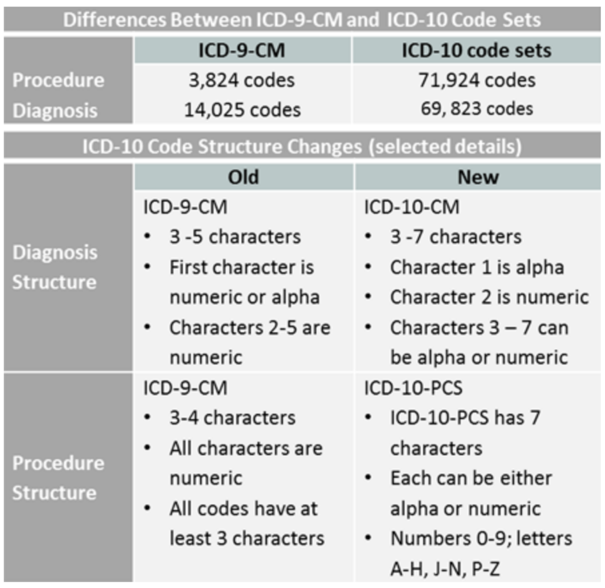
The International Classification of Diseases (ICD), approved by the World Health Organization (WHO), is a medical classification list of codes for diagnoses and procedures. ICD codes have been widely adopted by physicians and other healthcare providers for reimbursement, storage and retrieval of diagnostic information. Much of what is known about the extent, causes and consequences of human disease worldwide is based on the use of data classified according to ICD. It performs a variety of functions in a large part of the world (at least 120 countries) and has been translated into 43 languages [3]. ICD codes are generally separated between procedures codes (ICD-PCS) related to the *what* the physician has performed, and diagnostic codes (ICD-CM) related to the *why* he performed it.

Since their creation in 1948, ICD codes have evolved through periodic revisions. In the data that can be found at the time of writing for ML model training, we can encounter 3 different versions of the ICD codes:

- ICD-9 codes were created in 1979 and are found today only in legacy architectures. Roughly 18,000 codes are available in this classification [4].
- ICD-10 came into use in 1994. The main differences from previous ICD version are explained in figure 1.2. There are nearly 20 times as many procedure codes in ICD-10 than in ICD-9. There are nearly 5 times as many diagnosis codes in ICD-10 than in ICD-9. ICD-10 has alphanumeric categories instead of numeric ones. The order of some chapters have changed, some titles have been renamed, and pathologies have been grouped differently.
- ICD-11 has been approved in 2019 by the 72nd WHO assembly [3]. It is better suited to capture clinically relevant features of cases and to enable information to be summarized for a variety of purposes. In addition, it is flexible, allowing it to be used in more or less elaborate modes, and offers built-in support for several languages. It is also designed to ensure that data coded according to ICD-11 will be comparable with data coded according to ICD-10.

In addition, some countries have adapted the global ICD classification to their own needs, resulting in an even greater number of classifications. These include ICD-9-CM (USA), ICD-10-CM/PCS (USA), ICD-10-AM (Australia) and ICD-10-CA (Canada). One of the challenges of automated medicine is to take into account the evolution (deletion, addition, substitution, modification) of classifications that may coexist at the same time in electronic health records (EHR).

ICD classifications have a hierarchical structure. Characters at the beginning of the code give first-level indications, while those at the end give details of the disease, see 1.3. As such, ICD codes present a tree-like structure that can be leveraged.



Differences Between ICD-9-CM and ICD-10 Code Sets

	ICD-9-CM	ICD-10 code sets
Procedure	3,824 codes	71,924 codes
Diagnosis	14,025 codes	69, 823 codes

ICD-10 Code Structure Changes (selected details)

	Old	New
Diagnosis Structure	ICD-9-CM • 3 -5 characters • First character is numeric or alpha • Characters 2-5 are numeric	ICD-10-CM • 3 -7 characters • Character 1 is alpha • Character 2 is numeric • Characters 3 – 7 can be alpha or numeric
Procedure Structure	ICD-9-CM • 3-4 characters • All characters are numeric • All codes have at least 3 characters	ICD-10-PCS • ICD-10-PCS has 7 characters • Each can be either alpha or numeric • Numbers 0-9; letters A-H, J-N, P-Z

Figure 1.2: Main differences between ICD-9 and ICD-10. Source: National Center for Health Statistics

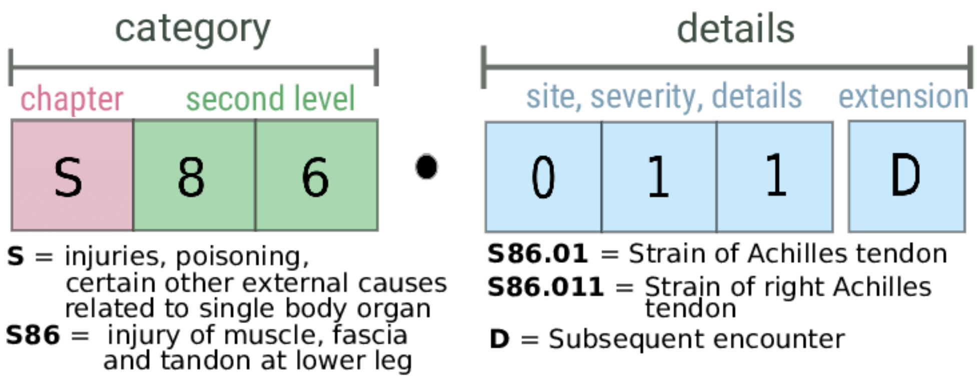

category
details
chapter
second level
site, severity, details
extension
S
8
6
•
0
1
1
D
S = injuries, poisoning, certain other external causes related to single body organ
S86 = injury of muscle, fascia and tandon at lower leg
S86.01 = Strain of Achilles tendon
S86.011 = Strain of right Achilles tendon
D = Subsequent encounter

Figure 1.3: ICD-10 code structure. Source:[5]

1.1.2 CPT codes

The Current Procedural Terminology (CPT) code set is a procedural code set developed by the American Medical Association (AMA). Unlike the procedure codes used in ICD, CPT codes are used for both inpatient and outpatient procedures.

CPT codes are composed of 5 digits numbers that exhibits a hierarchical structure. For instance codes in range 00100-01999 are related to anesthesia, and inside that range those in range 00100-00222 are related to head anesthesia. This decomposition goes on recursively.

Similarly to ICD, CPT codes have undergone many iterations. But unlike ICD, there are no public archives of the past versions. As such, it is harder to compare algorithms performance over time.

Although its use has been federally regulated in the USA, the CPT copyright has not fallen into the public domain. Users of the CPT code series must pay a license fee to the AMA. As a result, research interest has largely focused on ICD, to the detriment of CPT.

1.2 Medical coding in practice

Medical coding takes place every time a person consults a healthcare provider. The healthcare provider reviews the complaint and medical history, makes an expert assessment of what's wrong and how to treat the patient, and documents the visit. This documentation is not only the patient's permanent record, it's also how the healthcare provider is paid (although that part can vary considerably from place to place).

The patient's diagnosis, test results and treatment must be documented, not only for reimbursement purposes, but also to ensure high-quality care during subsequent visits. Personal information about a patient's health follows them through complaints and subsequent treatments, and must be easily understandable. This is particularly important when considering the hundreds of millions of visits, procedures and hospitalizations.

Medical coding requires a particular discipline. Medical codes must tell the whole story of the patient's encounter with the doctor, and be as accurate as possible to obtain reimbursement for services rendered. The difficulty lies in the fact that there are thousands of conditions, diseases, injuries and causes of death. There are also thousands of services provided by providers, and an equivalent number of drugs and injectable supplies to track. And in healthcare, there are multiple descriptions, acronyms, names and eponyms for every disease, procedure and tool.

1.3 The need for Automated Medical Coding

Since clinical coding is a non-trivial task for humans, automated clinical coding can be useful for several reasons. Firstly, the coding process typically involves many layers of abstraction or data summarization [6], for a variety of support (structured data, text, images). As such, training a professional medical coders is hard and the resulting coding is expensive. Second, coding is time-consuming. For instance in Scotland, a NHS medical coder usually codes about 60 cases a day (equivalent to 7–8min for each case). Even so, there is usually a backlog of cases to be coded, which can take several months or more (over a year) [7]. Third, manual coding is known to be prone to errors. Various reasons may account for that such as incompleteness in a patient's data, subjectivity in choosing the diagnosis codes, lack of coding expertise, or data entry errors [6]. The average accuracy of coding in the UK was around 83% with a large variance among studies (50-98%) [8].

According to statistics, the cost of coding errors and the financial investment devoted to improving coding quality are estimated at $25 billion a year in the United States [9].

Automated clinical coding is the idea that clinical coding may be automated by computers using AI techniques. There has been a surge of articles for automated clinical coding with deep learning (as the current mainstream approach of AI) in the last few years, more in chapter 3.

1.4 Goal

In this work, we pursue the goal of leveraging state-of-the art deep-learning and NLP techniques for automated medical coding on both ICD and CPT codes. Our own contribution is more detailed in chapters 3 and 4.

Besides, we believe that the hierarchy of ICD and CPT is a great asset for such a task that is often not taken in account. We will try leverage them using non-Euclidean geometry that is detailed in chapter 3. Still, non-Euclidean embeddings of medical codes goes further than automated medical coding and could also be used for fraud detection as proposed in chapter 5.

2 Datasets

Contents

2.1 MIMIC

MIMIC (Medical Information Mart for Intensive Care) is a large, freely-available database comprising deidentified health-related data from patients who were admitted to the critical care units of the Beth Israel Deaconess Medical Center in Boston, MA.

MIMIC supports a wide range of analytical studies covering epidemiology, improvement of clinical decision rules and development of digital tools. It has become mainstream in healthcare research for three reasons: it is freely available to researchers worldwide; it encompasses a diverse and very large population of intensive care unit (ICU) patients; and it contains highly granular data, including vital signs, laboratory results and medications.

Although access to the data is free, researchers must complete a recognized course on human research participant protection, which includes HIPAA (Health Insurance Portability and Accountability Act) requirements, and sign a Data Use Agreement, which outlines appropriate data use and security standards. Several versions of MIMIC have been released over the years, from MIMIC-II to the latest MIMIC-IV. All versions are stored and managed on PhysioNet's MIT servers [10].

In our work, we rely heavily on various MIMIC datasets. In this section, we propose to take a closer look at their composition and the way in which the data was collected. As the data supplied to the algorithm is absolutely crucial (garbage in-garbage out), it is essential to know where and how it was collected. Researcher Kate Crawford has shown that this aspect is often overlooked in AI research, and can lead to ethical problems [11].

2.1.1 MIMIC-III

MIMIC-III is a database associated with over forty thousand patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012 [12]. The database includes information such as demographics, vital sign measurements made at the bedside, laboratory test results, procedures, medications, caregiver notes, imaging reports and mortality (including post-hospital discharge).

The MIMIC-III database was populated with data acquired during routine hospital care, so there was no workload for caregivers or interference with their workflow. Data was downloaded from a number of sources, including the archives of critical care information systems, hospital electronic record databases and the Social Security Administration's Death Master File. Two different critical care information systems were in place during the data collection period: Philips CareVue Clinical Information System and iMDsoft MetaVision ICU. These systems were the source of clinical data [12].

The development of the MIMIC data model involved striking a balance between simplicity of interpretation and closeness to the ground truth. As such, the model reflects the underlying data sources, modified over the iterations of the MIMIC database in response to user feedback. Care was taken not to make any assumptions about the underlying data during the transformations, so that MIMIC-III faithfully represents the hospital's raw data.

Before the data was incorporated into the MIMIC-III database, it was first deidentified in accordance with HIPAA standards using structured data cleansing and date shifting [12]. The process of de-identifying structured data involved removing the eighteen identifiers listed in HIPAA, including fields such as patient name, phone number, address and dates. In particular, dates were randomly shifted into the future for each patient in a consistent manner to preserve intervals, resulting in stays taking place between the years 2100 and 2200. Time, day of week and approximate seasonality were preserved during date shifting. The dates of birth of patients aged over 89 were shifted to mask their real age and comply with HIPAA regulations: these patients appear in the database with an age of over 300 years.

As a result, MIMIC-III is a relational database consisting of 26 tables. However broadly speaking, 5 tables are used to define and track patient stays: ADMISSIONS, PATIENTS, ICUSTAYS, SERVICES, and TRANSFERS. The remaining tables contain data associated with patient care, such as physiological measurements, caregiver observations and billing information.

As for free-text notes, these can take a variety of forms: discharge summaries, nurse's observations and hospitalization histories. In line with HIPAA standards, protected health information has been removed from free text fields, using a rigorously evaluated de-identification system based on extensive dictionary searches and regular expression pattern matching [12]. For our purposes, we are mainly interested in the discharge summary and the corresponding ICD and CPT notes. For our use case, it should be noted that MIMIC-III contains only ICD-9 and CPT codes.

2.1.2 MIMIC-IV

MIMIC-IV builds upon the success of MIMIC-III, and incorporates numerous improvements over MIMIC-III. MIMIC-IV adopts a modular approach to data organization, highlighting data provenance and facilitating both individual and combined use of disparate data sources. MIMIC-IV is composed of data related to 300,000 patients for 430,000 admissions [13].

The creation of MIMIC-IV was carried out in three steps [13]:

- Acquisition: data of patients admitted to the emergency department or one of the intensive care units were extracted from the respective hospital databases. A master patient list was created containing all medical record numbers corresponding to patients admitted to an intensive care unit or emergency department between 2008 and 2019. All source tables were filtered on patient rows only in the main patient list.

- Preparation: the data were reorganized to better facilitate retrospective data analysis. This included denormalizing tables, removing audit trails, and reorganizing into fewer tables. The purpose of this process is to simplify the retrospective analysis of the database. It is important to note that no data cleaning steps were performed to ensure that the data reflects a real clinical data set.
- De-identification: patient identifiers as stipulated by HIPAA have been removed. Patient identifiers were replaced using a random number, resulting in anonymized integer identifiers for patients, hospitalizations, and intensive care stays. Structured data was filtered using lookup tables and allowlists. As in MIMIC-III, a free text deidentification algorithm was applied to remove personal information from free text. Each instance of personal information has been replaced with exactly three underscores. Finally, the date and time were randomly shifted into the future using an offset measured in days.

MIMIC-IV contains only text notes occurring within one year of a patient encounter, where an encounter is defined as an emergency room or hospital stay. Two categories of free text are included in the dataset: discharge summary and radiology report. Similar to MIMIC-III, we will look at medical codes and free text for our study. Nevertheless, MIMIC-IV contains both ICD-9 and ICD-10, but CPTs are not specified. This requires special attention when processing the data. In accordance with the novelty of MIMIC-IV, most of the models presented in chapter 3 are obtained on MIMIC-III and have not been tested on MIMIC-IV.

One of the big hurdles of MIMIC-III and IV for automated medical coding is the highly skewed distribution of codes frequencies that is displayed on figure 2.1. As such, predicting rarely used ICD or CPT is one of the significant difficulty of automated medical coding. This issue is exacerbated for code that is not even present in the original dataset. For instance, MIMIC-III only has less than 8,000 unique ICD-9 codes [14] compared to the 18,000 unique codes included in the full classification.

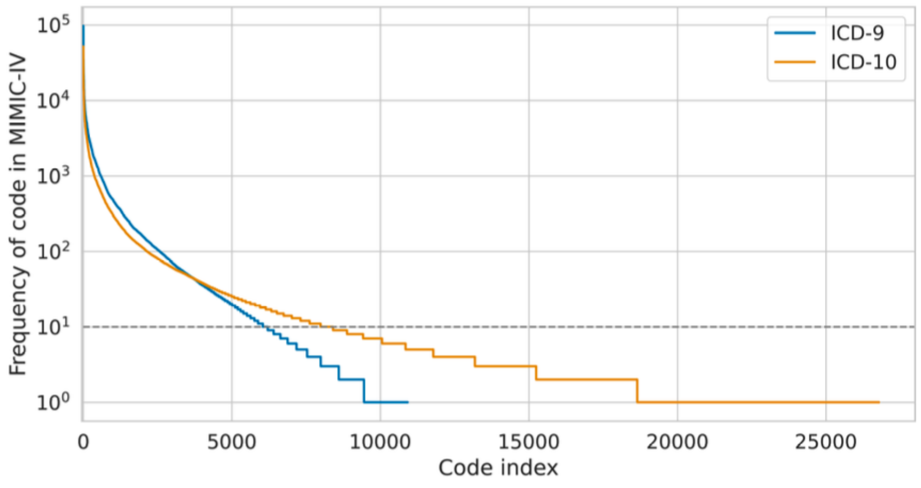


Figure 2.1: The frequency of ICD-9 and ICD-10 codes in MIMIC-IV before pre-processing. Source:[14]

2.1.3 Preprocessing

In 2018, the article by Mullenbach et al. [15] achieved state-of-the-art results for automated medical ICD coding on MIMIC-III through CNN and attention mechanism. Their code was open-sourced on Github and, as such, the preprocessing they performed was reused by most subsequent papers.

Their preprocessing consisted of lowercasing all text and removing words that only contain out-of-alphabet characters. Predicting procedures and diagnosis codes were treated as a single task. The dataset was split into training, validation, and test sets using random sampling, ensuring that no patient occurred in both the training and test set [15]. Models were evaluated using the micro and macro average of the area under the curve of the receiver operating characteristics (AUC-ROC), F1 score, and precision@k.

In 2023, the article by Edin et al. [14] spotted errors in their preprocessing and proposed a new way of managing it. They noticed non-stratified random sampling realised by Mullenbach et al. [15] lead to 54 % of the ICD codes in MIMIC-III not being sampled in the test set. This complicates the interpretation of results since these codes only contribute true negatives or false positives. Specifically, Edin et al [14] removed codes with fewer than 10 occurrences, doubled the test set size, and sampled the documents using multi-label stratified sampling. They ensured that no patient occurred in both the training and test set, preprocessed the text, and considered procedures and diagnosis codes as a single task as done by Mullenbach et al. [15].

For our work, we will rely on the processing performed by Edin et al [14]. Nonetheless their processing is done only on ICD codes, so we will adapt it for CPT codes as well. After preprocessing, the dataset from MIMIC-III is composed of 50,000 examples, where an example is a text sequence and the associated target codes, and the dataset from MIMIC-IV contains 120,000 examples.

2.2 Qantev's partner data

In addition to MIMIC data, freely available to researchers, we have access to specific data from insurers and partners with whom Qantev works.

The data provided by insurers are generally scans of medical reimbursement forms. To make it usable for our task, we perform various tasks. We detect and analyze the text on these forms using our custom OCR (Optical Character Recognition) designed in-house. We then select the most important areas for our task, such as diagnosis, treatment and cause of the disease. If the identified text is not english, we translate it using the python *deep-translator* library. We then format the data in the same way as the MIMIC dataset with the same processing applied to the text (lowercase letters, word deletion, ..).

In the end, data from insurers represent 937 examples. This is way less than in MIMIC-III and MIMIC-IV but we will see in chapter 4 their impact is not negligible. Let us also clarify that all ICD codes in this dataset are ICD-10 and that for each text note we only have one corresponding ICD. This is different from MIMIC data where for a clinical note we have different codes, generally between 5 and 20. Finally, the size of the note is also different: in the insurer data we have on average 70 words per note while we have 1,500 words per note in the MIMIC clinical notes.

3 Background on Automated Medical Coding

Contents

3.1 Background on NLP

Natural Language Processing (NLP) has undergone a remarkable transformation in recent years, primarily driven by the advent of deep learning techniques, with Transformers emerging as a pivotal architectural innovation. This background section provides a comprehensive overview of the evolution of NLP, emphasizing the critical role of Transformers, and highlights key milestones in the field. Throughout, we draw upon seminal works and breakthroughs to elucidate the rich tapestry of NLP research.

Prior to the deep learning era, NLP heavily relied on handcrafted linguistic rules and statistical models. The introduction of word embeddings, notably Word2vec by Mikolov et al. "Efficient Estimation of Word Representations in Vector Space" [16] marked a significant shift towards deep learning in NLP. Combined with recurrent neural networks (RNN) [17] and long short-term memory networks (LSTM) [18], Word2Vec played a pivotal role in modeling natural language.

During the early 2010s, RNNs and LSTMs found applications in various NLP tasks, such as machine translation, speech recognition, and sentiment analysis. "Sequence to Sequence Learning with Neural Networks" by Sutskever et al. [19] demonstrated the effectiveness of LSTMs in machine translation. Despite their initial successes, RNNs and LSTMs faced challenges with scalability and parallelization.

Transformers, as introduced in ”Attention Is All You Need” by Vaswani et al. [20], represent a pivotal moment in NLP. Their self-attention mechanism revolutionized sequence-to-sequence tasks, enabling the modeling of long-range dependencies efficiently. As a result, they gradually supplanted RNN-based architectures as the go-to choice for NLP researchers. This architectural shift enabled the development of models like GPT (Generative Pre-trained Transformer) series and the BERT embeddings, reshaping the NLP landscape. Specifically, Google's BERT (Bidirectional Encoder Representations from Transformers) as detailed in ”BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” by Devlin et al. [21] laid the foundation for large-scale pre-trained language models (PLM).

The impact of Transformers extends beyond research. They are now commonplace in wide range of industry and beyond NLP, since they are now widely used in computer vision as well. Still, despite their successes, Transformers have inherent limitations, such as their substantial computational demands, sensitivity to input phrasing, and struggles with commonsense reasoning.

3.2 Background on hyperbolic embeddings

Learning representations of symbolic data such as text, graphs and multi-relational data has become a central paradigm in machine learning. For instance, word embeddings such as Word2Vec [16], Glove [22] and FastText [23] are widely used for tasks ranging from machine translation to sentiment analysis. Typically, the goal of an embedding method is to organize symbolic objects (e.g. words, entities, concepts) such that their similarity or distance in the embedding space reflects their semantic similarity. Although embedding methods have proven effective in many applications, they suffer from a fundamental limitation: their ability to model complex models is inherently limited by the dimensionality of the embedding space. However, Nickel and Kiela [24] showed that for graph that present and underlying tree-like structure more efficient representations can be obtained, not in Euclidean but in hyperbolic space, i.e. space with constant negative curvature. Informally, hyperbolic space can be thought of as a continuous version of trees and as such it is naturally equipped to model hierarchical structures. For instance, it has been shown that any finite tree can be embedded into a finite hyperbolic space such that distances are preserved approximately [25]. In contrast, Bourgain's theorem [26] shows that Euclidean space cannot achieve comparably low distortion for trees, even using an unbounded number of dimensions. Below we propose a quick introduction to hyperbolic embeddings.

3.2.1 Geometry concepts

A manifold is a topological space (set with collection of open subsets) that looks locally Euclidean, i.e. for each open subset, we can find a map (called a coordinate map) that maps the subset to an Euclidean space. For instance, a sphere is 2-D manifold than can be locally maps to $\mathbb{R}^2$. The description of most manifolds requires more than one chart. A specific collection of charts which covers a manifold is called an atlas. Charts in an atlas may overlap and a single point of a manifold may be represented in several charts, just as a map of Europe and a map of France may both contain Paris. Given two overlapping charts, a transition function can be defined which goes from an open ball in $\mathbb{R}^n$ to the manifold and then back to another (or perhaps the same) open ball in $\mathbb{R}^n$. The resultant map, a change of coordinates, is called a transition map. On figure 3.1 the open subsets are noted U, the coordinate map φ and the transition map τ. A smooth manifold is a manifold for which transition maps are smooth.

Now that we have defined a manifold, we can define what is a tangent space. The tangent space of a manifold is a generalization of tangent lines to curves in two-dimensional space and tangent planes to surfaces in three-dimensional space. On manifold, we can define curve that are smooth path $\gamma : [0, 1] \to M$. The velocity of a point $p \in M$ on a particular curve defines a tangent vector, and if we take all the tangent vectors of all curves at point p we get the tangent space T_pM. T_pM

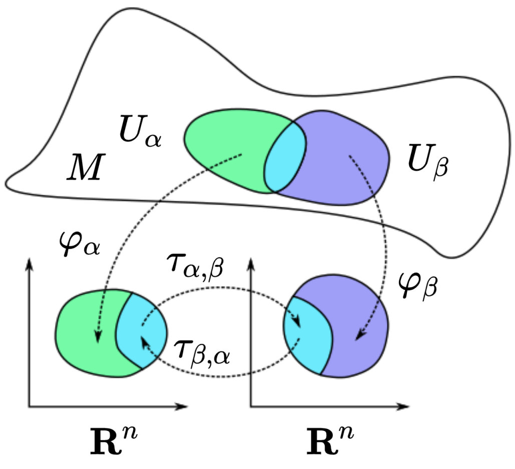


Figure 3.1: Two charts on a manifold, and their respective transition map. Source: Wikipedia Atlas

has the same dimension as the manifold and has an Euclidean structure. This Euclidean structures is of primary importance to define optimisation algorithm on manifolds [27].

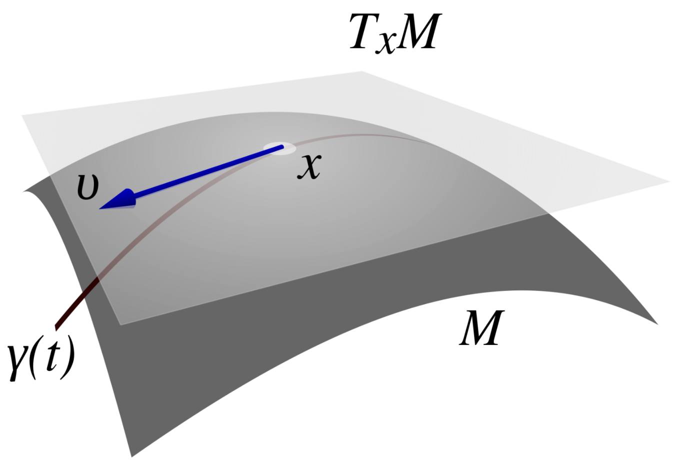


Figure 3.2: The tangent space and a tangent vector, along a curve traveling through. Source: Wikipedia Tangent Space

Let's now define a Riemannian manifold. Let M be a smooth manifold, $p \in M$ be a point, and T_pM be the tangent space to the point p. If M is equipped with a Riemannian metric $g_p : T_pM \times T_pM \to \mathbb{R}^+$, then the pair (M, g) is called a Riemannian manifold. The Riemannian metric, a smoothly varying function that induces geometric notions such as length and angle by defining an inner product on the tangent space, is used to do calculus on manifolds. In particular, the shortest-distance paths on manifolds are called geodesics. To compute distance functions on a Riemannian manifold, the metric tensor g is integrated along the geodesic. For example, the norm of $v \in T_pM$ is defined as $\|v\|_g = g_p(v,v)^{\frac{1}{2}}$. In Euclidean space $\mathbb{R}^d$, each tangent space $Tp\mathbb{R}^d$ is canonically identified with $\mathbb{R}^d$, and the metric tensor g^E is simply the normal inner product.

Finally, we call exponential map the function that maps a point from the tangent space to the manifold $exp_p : T_pM \to M$ and the logarithmic map the function that maps a point from the manifold to the tangent space $log_p : M \to T_pM$. For some particular manifolds structure, we have closed-form equation for these functions.

3.2.2 Hyperbolic geometry

Non-Euclidean geometry was developed in the 19th century. It arises by replacing the parallel postulate, which states that given a line and a point not on it, it exists exactly one line parallel to the given line that can be drawn through the point. In spherical geometry no parallel lines exists, while in hyperbolic geometry an infinite number of parallel line exist.

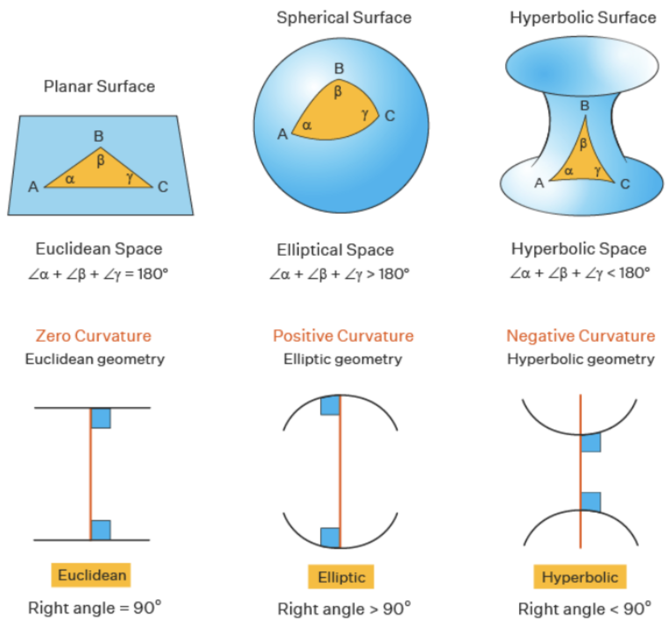


Figure 3.3: Different type of geometry. Source: Cuemath

As such, hyperbolic geometry is a non-Euclidean geometry which studies spaces of constant negative curvature. Hyperbolic space are useful in the sense that they can be used as continuous version of trees. A regular tree with branching factor b has $(b+1)b^{l-1}$ nodes at level l. Hence, the number of children grows exponentially with their distance to the root of the tree. In hyperbolic geometry this kind of tree structure can be modeled easily in two dimensions: nodes that are exactly l levels below the root are placed on a sphere in hyperbolic space with radius $r \propto l$ and nodes that are less than l levels below the root are located within this sphere. This type of construction is possible as hyperbolic disc area and circle length grow exponentially with their radius. In $\mathbb{R}^2$, a similar construction is not possible as circle length $(2\pi r)$ and disc area (πr^2) grow only linearly and quadratically with regard to r in Euclidean geometry.

3.3 Background on applications to medical codes

In recent years, a lot of interest has been granted to automated medical coding based on the breakthrough of deep learning techniques. Usually, this problem is framed as a multi-class multi-label text classification model which means that for one document the model predicts various codes [2]. The usual workflow of models is composed of an encoder to extract text features and a decoder for code prediction. Some models also leverage auxiliary data such as code descriptions, code descriptions or medical taxonomies.

Several recent work attempted to approach this task with neural networks. Choi et al. [28] and Baumel et al. [29] used RNN to encode the EHR data for predicting diagnostic results. Mullenbach et al. [15] used CNN as encoder. Besides, they introduced an attention decoder

(Label-wise Attention Network) instead of a direct fully connected layer to that is better suited for extreme multi-label tasks. Also, Tsai et al. [30] introduced various ways of leveraging the hierarchical knowledge of ICD by adding refined loss functions. Vu et al. created LAAT [31] in which they integrated a bidirectional LSTM with an improved label-aware attention mechanism as decoder. Recently, Cao et al. [32] introduced HyperCore, in which they proposed to train ICD code embeddings in hyperbolic space to model the hierarchical structure. Additionally, they used graph neural network to capture the code co-occurrences. To the best of our knowledge, current state-of-the art is reached by PLM-ICD implemented by Hua et al. [33] in 2022. They rely on BERT-like encoder (whereas others models mainly rely on Word2Vec embeddings) and use the same decoder as in LAAT. With this structure they were able to reach a micro-f1 of 50.4% on the full MIMIC-III test set. Others works that explore different encoders include the one of Zhang et al. [34] where they used BERT-like architecture that trained from scratch on longer context lenght (1,024).

In all cases, research has been faced with the following challenges:

- Noisy and lengthy clinical notes: they contain specific medical vocabulary, non-standards abbrevaiation and often misspellings. Depending on the context they can be long (e.g. MIMIC) and contain various types of information.
- High-dimensional medical codes: as the label set is large, this problem is usually considered an extreme classification problem. As explained in chapter 2, classifications are usually made of at least tens of thousands codes.
- Uneven distribution of codes: Due to the existence of common and rare diseases, the distribution of medical codes in an EHR system is unbalanced, also known as the long tail phenomenon. As MIMIC focuses on intensive care units, it aims to be as broad as possible in the codes collected. Still, a comprehensive dataset for automated classification seems out of sight for the foreseeable future.

3.4 Own contribution

As we have seen, most recent models focused on either developing an effective interaction between note representations and code representations ([15], [32]) or focusing on the choice of the note encoder while not leveraging codes auxiliary data ([34], [33]).

Our goal in this work is to reconcile the two approaches. We will build on the work done on PLM-ICD and try to incorporate codes hyperbolic hierarchy in the decoder to enhance the model. As we will in following sections, this initial goal was not reached because of instabilities in codes hyperbolic embeddings (see chapter 5). Nonetheless, based on the hierarchy of codes that we tried to leverage in hyperbolic embedding, we propose a new hierarchical decoder that enhances PLM-ICD capabilities (see chapter 4).

4 Automated Medical Coding

Contents

4.1 PLM hierarchical framework

The task of medical code prediction is formulated as a multi-label classification problem [15]. Given a clinical note of $|d|$ tokens $\mathbf{d} = \left\{t_1, t_2, \cdots, t_{|d|}\right\}$ in EHR, the goal is to predict a set of medical codes $\mathbf{y} \subseteq \mathcal{Y}$, where $\mathcal{Y}$ denotes the set of all possible codes. Typically, the labels are represented as a binary vector $\mathbf{y} \in \{0,1\}^{|\mathcal{Y}|}$, where each bit y_i indicates whether the corresponding label is presented in the note. Here, we present the original PLM-ICD framework taken from [33] and our own improvement, the hierarchical embedding.

4.1.1 PLM : Pretrained Language Model with segment pooling

Automated coding is a domain-specific task where the input text consists of clinical notes written by clinicians. The clinical notes contain many biomedical terms, and understanding these terms is essential in order to assign ICD codes accurately. While general PLMs are pretrained on large amount of text, the pretraining corpora usually does not contain biomedical text, not to mention clinical records. In order to mitigate the domain mismatch problem, we propose to utilize the PLMs that are pretrained on biomedical and clinical text, e.g. BioBERT [35], PubMedBERT [36], and RoBERTa-PM [37]. These PLMs are specifically pretrained for biomedical tasks and proven to be effective on various downstream tasks. We can plug-and-play the domain specific PLMs since their architectural design and pretraining objective are identical to their general domain counterparts. This makes our framework agnostic to the type of PLMs, i.e. we can apply any transformer-based

PLMs as the encoder. For this work, we will stick to RoBERTa-PM which is the one used in the original PLM-ICD paper.

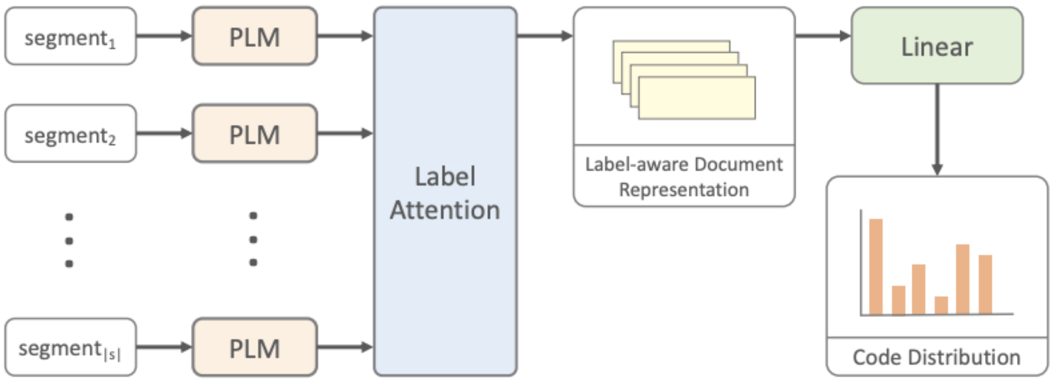


Figure 4.1: PLM-ICD model. Source: [33]

The problem with using a transformer encoder is that we are limited in the number of input tokens. For example, RoBERTa-PM has a maximum input size of 512 tokens, which is below the average token count of the MIMIC clinical notes, which is 1500 to 2000 tokens.

In order to tackle the long input text problem, segment pooling is used to surpass the maximum length limitation of PLMs. The segment pooling mechanism first splits the whole document into segments that are shorter than the maximum length, and encodes them into segment representations with PLMs. After encoding segments, the segment representations are aggregated as the representations for the full document.

More formally, given a document $d = \left\{t_1, t_2, \ldots, t_{|d|}\right\}$ of $|d|$ tokens, we split it into $|s|$ consecutive segments s_i of c tokens:

$$s_i = \{t_j \mid c \cdot i \leq j < c \cdot (i+1)\}$$

The segments are fed into PLMs separately to compute hidden representations, then concatenated to obtain the hidden representations of all tokens:

$$\mathbf{H} = \text{concat}\left(PLM\left(s_1\right), \cdots, PLM\left(s_{|s|}\right)\right) \tag{4.1}$$

The token-wise hidden representations $\mathbf{H}$ can then be used to make prediction based on the whole document.

Note that the token-wise hidden representations $\mathbf{H} \in \mathbb{R}^{|d| \times d_e}$ where $|d|$ is the number of tokens in input and d_e corresponds to the size of RoBERTa-PM's embedding space, i.e. 768 in our case.

4.1.2 Label wise attention

To combat the problem of a large set of labels, PLM-ICD rely on the label-attentive attention mechanism proposed by [31] to learn label-specific representations that capture important text fragments relevant to certain labels. After the token-wise hidden representations $\mathbf{H}$ are obtained, the goal is to transform $\mathbf{H}$ into label-specific representations with attention mechanism.

The label-aware attention takes $\mathbf{H}$ as input and compute $|\mathcal{Y}|$ label-specific representations. This mechanism can be formulated into 2 steps. First, a label-wise attention weight matrix $\mathbf{A}$ is computed as:

$$\begin{aligned} \mathbf{Z} &= \tanh(\mathbf{HV}) \\ \mathbf{A} &= \text{softmax}(\mathbf{WZ}^\top) \end{aligned} \tag{4.2}$$

where $\mathbf{V}$ and $\mathbf{W}$ are linear transforms.

The first linear transformation $\mathbf{V} \in \mathbb{R}^{d_e \times d_e}$ is used to mix the output information of each RoBERTa-PM' sequence. In fact, the PLM output embedding of the words only has the information inside the segment. The second transformation $\mathbf{W} \in \mathbb{R}^{|\mathbf{L}| \times d_e}$ where $|\mathbf{L}|$ is the number of medical codes to predict, so each row of this matrix corresponds to a medical code and can be interpreted as the embeddings to be learned from each medical code. Thus the product $\mathbf{W}\mathbf{Z}^{\top}$ is the similarity score between the embedding matrix of the document and those of each of the labels.

The i^{th} row of $\mathbf{A}$ represents the weights of the i^{th} label, and the softmax function is performed for each label to form a distribution over all tokens. Then, the matrix $\mathbf{A}$ is used to perform a weighted-sum of $\mathbf{H}$ to compute the label-specific document representation:

$$\mathbf{D} = \mathbf{A}\mathbf{H} \tag{4.3}$$

where the row $\mathbf{D}_i$ represents the document representations for the i^{th} label.

Finally, we use the label-specific document representation $\mathbf{D}$ to make predictions:

$$\mathbf{p}_i = \text{sigmoid}\left(\langle \mathbf{L}_i, \mathbf{D}_i \rangle\right) \tag{4.4}$$

where $\mathbf{L}_i$ is a vector for the i^{th} label, $\langle \cdot \rangle$ represents inner product between two vectors, $\mathbf{p}_i$ is the predicted probability of the i^{th} label. Note that the inner product could also be seen as a linear transform with output size 1. We can then assign labels to a document based on a predefined threshold t. The training objective is to minimize the binary cross-entropy loss $\mathcal{L}(\mathbf{y}, \mathbf{p})$:

$$-\frac{1}{|\mathbf{y}|} \sum_{i=1}^{|\mathbf{y}|} \left(\mathbf{y}_i \log \mathbf{p}_i + (1 - \mathbf{y}_i) \log\left(1 - \mathbf{p}_i\right)\right) \tag{4.5}$$

It's interesting to note that the loss chosen here is binary cross-entropy and not categorical cross-entropy, because the problem is formulated as a multi-class multi-label problem, i.e. each note has several labels and the number of labels is not defined in advance. Thus, $\mathbf{p}_i$ is the predicted probability for label i^{th}. In inference, as well as in the calculation of metrics during learning, we choose the threshold t for accepting or rejecting the label. The threshold is chosen to maximize the f1-micro score on the validation set.

4.1.3 Hierarchical embedding

Our most important contribution to the existing PLM-ICD model is the introduction of hierarchical embedding. Indeed, previous model introduces an embedding for each label in the matrix $\mathbf{W} \in \mathbb{R}^{|\mathbf{L}| \times d_e}$. These embeddings are initialized randomly (Gaussian vectors) and then learned as training progresses. Nonetheless, along the same lines than hyperbolic embeddings detailed in chapter 5, we could leveraged the code hierarchy to obtain hierarchical embeddings.

The aim is then to find a module to capture the hierarchy of medical codes in embeddings. For these embeddings to be coherent, we need the following conditions:

- The tree structure must be respected, i.e 2 codes with the same root must be more similar than 2 codes from different groups.
- We want the position of the characters to be important: a 9 in the first position must not have any correlation with a 9 in a second position, and if the parents of two 9s are not identical, then there must be no correlation in the way these 9s are embedded.
- The complexity of this new module should be of the same order of magnitude as the original label wise attention.

Let $t_j = X_1X_2$ and $t_j = X_1X_3$ be two medical codes with X_1 the common part of the 2 codes, X_2 and X_3 the part that differs. Let $f_\theta : |d| \to \mathbb{R}$ be the function that assigns each code to its embedding, then there must exist 3 functions g_θ,h_θ and z such that:

$$\begin{aligned} f_\theta(t_i) &= z(g_\theta(X_1), h_\theta(X_1X_2)) \\ f_\theta(t_j) &= z(g_\theta(X_1), h_\theta(X_1X_3)) \end{aligned} \tag{4.6}$$

In the following, we'll use the operation + for z. Besides we will use the various levels where the code match.

To make it clear let's take a concrete example with CPT codes, but this process can be applied to any hierarchy. If we take the CPTs $t_i = 99764$ and $t_j = 99864$, then the previous condition gives us the following implications:

- $g_\theta(9_i) = g_\theta(9_j)$
- $g_\theta(99_i) = g_\theta(99_j)$
- $f_\theta(t_i) = g_\theta(9) + g_\theta(99) + g_\theta(997) + g_\theta(9976) + g_\theta(99764)$
- $f_\theta(t_j) = g_\theta(9) + g_\theta(99) + g_\theta(998) + g_\theta(9986) + g_\theta(99864)$

To add that in the previously existing decoder module, we process as follow. Let $|t|$ be the number of different CPTs, $|t_4|$ the number of possible 4-digit cpt roots (for example: 9975 for 99751 and 99752 ...), $|t_3|$ the number of possible 3-digit roots, $|t_2|$ the number of possible 2-digit roots, $|t_1|$ the number of possible 1-digit roots.

Let $A \in \mathbb{R}^{|t|\times d_e}$, $A_4 \in \mathbb{R}^{|t_4|\times d_e}$, $A_3 \in \mathbb{R}^{|t_3|\times d_e}$, $A_2 \in \mathbb{R}^{|t_2|\times d_e}$ and $A_1 \in \mathbb{R}^{|t_1|\times d_e}$ be randomly initialized matrixes that will be the embedding matrices for each node in the tree. Then, to respect the last algorithm, it's enough to take W as:

$$W = PA + P_4A_4 + P_3A_3 + P_2A_2 + P_1A_1 \tag{4.7}$$

where P, P_4, P_3, P_2, P_1 are matrices used to duplicate rows in order to comply with the dimensions. For instance, $P_2 \in \mathbb{R}^{|t|\times|t_2|}$. In particular:

$$(P_l)_{i,j} = \begin{cases} 1 & if\ \sigma_l(j) = \sigma(i)[0:l] \\ 0 & otherwise \end{cases} \tag{4.8}$$

where σ is the function which, starting from the index of all codes, gives the corresponding code and σ_l is the function which, starting from the index of all l-digits nodes ($|t_l|$), gives the corresponding l-digits node.

It's interesting to note that this method has about the same memory complexity as the original label wise attention algorithm; we store $m \approx |t|(1 + \frac{1}{10} + \frac{1}{100} + \frac{1}{1000} + \frac{1}{10000})$ embeddings instead of $|t|$ embeddings.

4.2 Experiments

4.2.1 Practical settings

Several experiments were carried out on the datasets presented in chapter 2.

We carried out experiments on the ICD-9 codes of the MIMIC-III dataset using both versions of the algorithm, namely the original PLM-ICD and our hierarchical version, called PLM-ICD hierarchical. The model was also trained on CPT codes, making it one of the first models trained on CPT codes (we didn't find any literature applying automated medical coding on CPT codes). We'll call these two versions PLM-CPT and PLM-CPT hierarchical.

The model was also trained on the MIMIC-IV dataset for ICD-10 codes in both its versions. In addition, it was trained on a concatenated dataset comprising MIMIC-IV and Qantev's partners data. This approach is akin to fine-tuning on the smallest partner dataset. It was not possible to fine-tune the model directly, as it needs to know in advance which labels are present in the dataset. However, the partners dataset contains ICD-10 codes that are not present in the MIMIC-IV dataset.

In addition, we divided the dataset into training, validation and test sets with the following ratios: 0.75 for training, 0.1 for validation and 0.15 for test. To better address the multi-label classification scenario, we used the iterative stratification algorithm to divide the datasets.

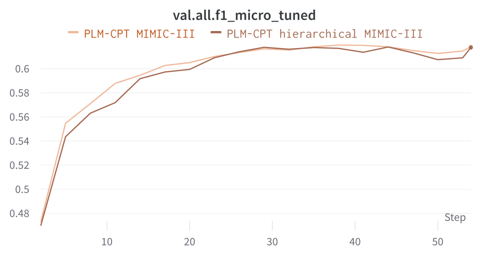
Training was performed for around 20 epochs (sometimes less to avoid excessive GPU utilization), and a linear decay of the learning rate was applied. We used Adam for optimization with a target learning rate of 5.10^{-5}. For segment pooling, we split the text into sequences of 128 tokens. In addition, we used a single Tesla v100 GPU for training. Due to the GPU's RAM limitation, we had to limit our batch size to 1. For tracking experiments and metrics, we used Weights&Biases.

4.2.2 Results

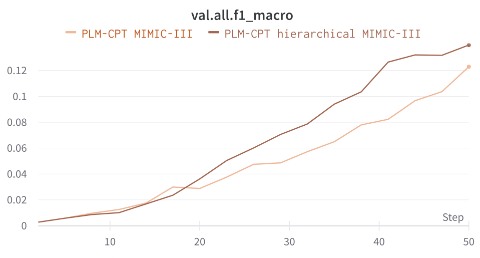
Global results for all models and all training set are available on table 4.1.

PLM-CPT vs PLM-CPT hierarchical, on MIMIC-III

We represent f1 macro and f1 micro as the problem is multi-label classification and the micro results do not take into account the imbalance class. Note that 3 steps represents one epoch. Results are visible on figure 4.2. As expected, we obtain similar performances for the micro scores, which shows that our hierarchical model remains at the level of the original model that obtained the best results on automated medical coding benchmarks. What's more, for the hierarchical model, we can see that there is a clear improvement for f1 macro. This improvement on macro score is not negligible for medical codes, as the large number of labels is recognized difficulties in the field.



(a) f1 micro on validation set over epochs

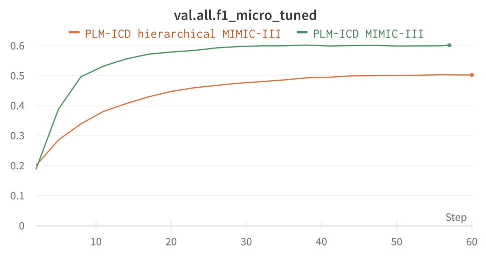


(b) f1 macro on validation set over epochs

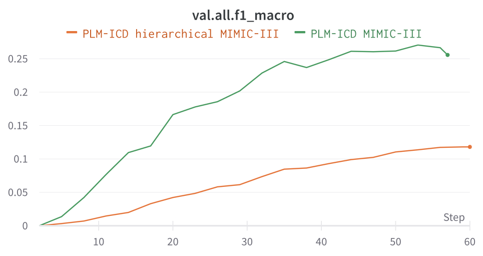
Figure 4.2: CPT coding results on MIMIC-III

PLM-ICD vs PLM-ICD hierarchical, on MIMIC-III

Results are visible on figure 4.3. This time, the hierarchical model performed less well for both metrics. This may be explained by the particular classification of ICD-9, which is a little different from ICD-10 and a little less structured. To take this into account, we tried a simpler customized hierarchy, but it seems that this was not sufficient for our model to perform well. Nevertheless, this proves that the hierarchical decoder cannot be applied to all classifications, whatever their structure.



(a) f1 micro on validation set over epochs



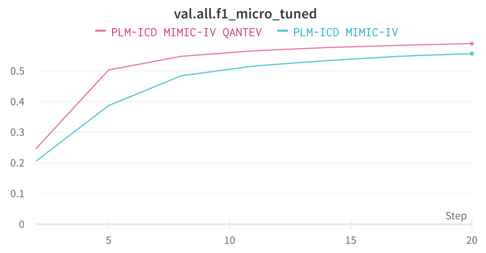
(b) f1 macro on validation set over epochs

Figure 4.3: ICD-9 coding results on MIMIC-III

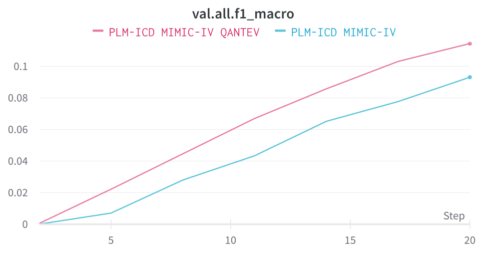
PLM-ICD on MIMIC-IV vs PLM-ICD on MIMIC-IV + Qantev data

Here, we do not seek to compare the hierarchical model with the original PLM-ICD. Instead, we analyze whether adding more data to MIMIC can bring better results. The results are shown in figure 4.4. Interestingly, it seems that adding Qantev's partners data to MIMIC-IV brings better results. This may be difficult to explain, as it is highly data-dependent. Most likely, the shorter text sequences encountered in the additional Qantev's partner data, together with the frugality of the codes used per text sequence, allow the model to be more robust.

In addition, it seems that not all metrics reach a plateau, so it might be interesting to run a longer training or a training with more GPUs to see if this further increases the results.



(a) f1 micro on validation set over epochs

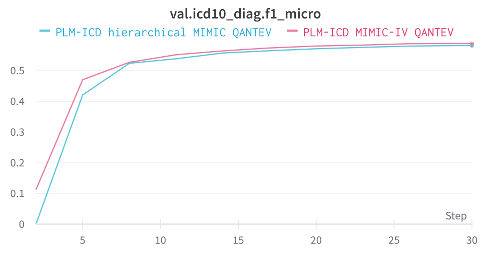


(b) f1 macro on validation set over epochs

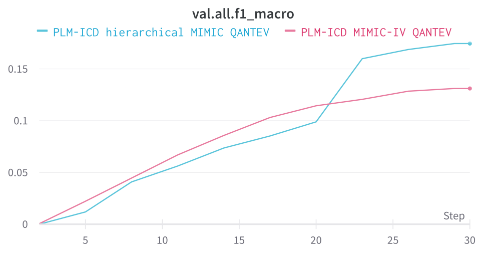
Figure 4.4: ICD-10 coding results on MIMIC-IV and MIMIC-IV + Qantev data

PLM-ICD vs PLM-ICD hierarchical, on MIMIC-IV + Qantev data

As we have seen that using Qantev data in conjunction with MIMIC-IV brings better results, we will stick to it for last experiment. Results are displayed on figure 4.5. This time we have better results with the hierarchical model. This may be explained by the ICD-10 structure which is more structured than ICD-9. Interestingly, the hierarchical model doesn't produce better macro results directly, but only after a few epochs. This underlines the greater training complexity of this model.



(a) f1 micro on validation set over epochs



(b) f1 macro on validation set over epochs

Figure 4.5: ICD-10 coding results on MIMIC-IV + Qantev data

Model	AUC-ROC	precision micro	precision macro	f1 micro	f1 macro
PLM-CPT, MIMIC-III	0.9964	0.587	0.081	0.588	0.069
PLM-CPT hierarchical, MIMIC-III	0.9957	0.583	0.116	0.612	0.116
PLM-ICD, MIMIC III	0.9896	0.609	0.293	0.596	0.264
PLM-ICD hierarchical, MIMIC-III	0.9764	0.528	0.171	0.501	0.152
PLM-ICD, MIMIC-IV	0.9912	0.616	0.152	0.574	0.129
PLM-ICD, MIMIC-IV + Qantev data	0.9918	0.631	0.173	0.587	0.151
PLM-ICD hierarchical, MIMIC-IV + Qantev data	0.9919	0.622	0.276	0.569	0.219

Table 4.1: Global results on test set for all models and training set

5 Hyperbolic embedding

Contents

5.1 Poincaré model

There exists various models of hyperbolic geometry (Hyperboloid, Beltrami-Klein, Poincaré half-plane) that are all mathematically equivalent. For our study we will use Poincaré ball model, as it is well-suited for gradient-based optimization, that we detail below.

Let $B^d = \{x \in \mathbb{R}^d, \|x\| < 1\}$ be the open d-dimensional unit ball, where $\|\cdot\|$ denotes the Euclidean norm. The Poincaré ball model of hyperbolic space corresponds to the Riemannian manifold (B^d, g_x), i.e. the open unit ball equipped with the Riemannian metric tensor $g_x = (\frac{2}{1-\|x\|^2})^2 g^E$ where $x \in B^d$ and g^E denotes the Euclidean metric tensor. Furthermore, the distance between two points $u, v \in B^d$ is given by :

$$d(u,v) = \operatorname{arccosh}\left(1 + 2\frac{\|u-v\|^2}{(1-\|u\|^2)(1-\|v\|^2)}\right) \tag{5.1}$$

Geodesics in B^d are circles that are orthogonal to the boundary. Due to the negative curvature of the Poincaré disk in 2 dimensions, the distance of points increases exponentially, relative to their Euclidean distance, the closer they are to the boundary. Embedding of a regular tree is displayed on figure 5.1 such that all connected nodes are spaced equally far apart (i.e., all black line segments have identical hyperbolic length).

It can be seen from equation 5.1, that the distance within the Poincaré ball changes smoothly with respect to the location of u and v. This locality property of the Poincaré distance is key for

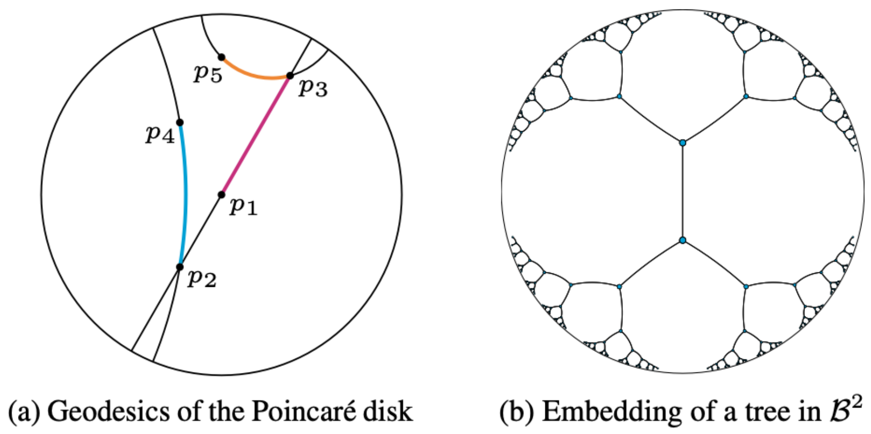

p_5
p_3
p_4
p_1
p_2
(a) Geodesics of the Poincaré disk
(b) Embedding of a tree in $\mathcal{B}^2$

Figure 5.1: Geometry in the Poincaré disk. Source: [24]

finding continuous embeddings of hierarchies. Furthermore, it is symmetric and that the hierarchical organization of the space is solely determined by the distance of points to the origin.

Let's note the Poincaré unit ball corresponds to an hyperbolic space of curvature -1. However, it can be adapted for other curvatures. If we want to emulate an hyperbolic space of curvature K, then we need to do the following modifications:

$$R = \frac{1}{\sqrt{-K}} \tag{5.2}$$

$$d_K(u,v) = R \times d(\frac{u}{R}, \frac{v}{R}) \tag{5.3}$$

$$g_x = (\frac{2}{1 - \|\frac{x}{R}\|^2})^2 g^E \tag{5.4}$$

$$\tag{5.5}$$

5.2 Optimisation

We build on the framework presented in [24]. To compute Poincaré embeddings for a set of symbols S, we are then interested in finding embeddings $\Theta = \{\theta_i\}_{i=1}^n$, where $\theta_i \in B^d$ is the embedding of symbol S_i. We assume we are given a problem-specific loss function $L(\Theta)$ which encourages semantically similar objects to be close in the embedding space according to their Poincaré distance. To estimate θ, we then solve the optimization problem:

$$\Theta' = \arg\min L(\Theta) \text{ s.t. } \forall \theta_i \in \Theta : \|\theta_i\| < 1 \tag{5.6}$$

Since the Poincaré ball has a Riemannian manifold structure, we can optimize Equation 5.6 via stochastic Riemannian optimization methods such as RSGD [27]. In particular, let $T_\theta B$ denote the tangent space of a point $\theta \in B^d$. Furthermore, let $\nabla_R \in T_\theta B$ denote the Riemannian gradient of $L(\theta)$ and let ∇_E denote the Euclidean gradient of $L(\theta)$. Using RSGD, parameter updates to minimize Equation 5.6 are then of the form:

$$\theta_{t+1} = exp_{\theta_t}\big(-\eta_t \nabla_R L(\theta_t)\big) \tag{5.7}$$

where exp_{θ_t} denotes the exponential map onto B at θ_t and η_t is the learning rate at time t.

To derive the Riemannian gradient from the Euclidean gradient, it is sufficient to rescale ∇_E with the inverse of the Poincaré ball metric tensor, i.e., g_θ^{-1}. Since g_θ is a scalar matrix, the inverse is trivial to compute. Furthermore, since Equation 5.1 is fully differentiable, the Euclidean gradient can easily be derived using standard calculus. As retraction method, we use the 1st order approximation of exponential map $exp_\theta(v) = \theta + v$ which corresponds to the Euclidean approximation and usually brings better results than the exact operation. In summary, the full update for a single embedding is then of the form:

$$\theta_{t+1} = \theta_t - \eta_t \frac{(1 - \|\theta_t\|^2)^2}{4} \nabla_E \tag{5.8}$$

Mathematically, an embedding is a mapping $f : U \to V$ for spaces U,V with distances d_U,d_V. To evaluate the quality of embeddings either we have a downstream task, either we use some ad-hoc metrics to quantify the preservation of the graph in continuous space. In the later case, it is usual to work with two metrics. The first one is the distortion D. For an n points embedding,

$$D(f) = \frac{1}{\binom{n}{2}} \left(\sum_{u,v \in U : u \neq v} \frac{|d_V(f(u), f(v)) - d_U(u,v)|}{d_U(u,v)} \right) \tag{5.9}$$

The best distortion is $D(f) = 0$, preserving the edge lengths exactly. This is a global metric, as it depends directly on the underlying distances rather than the local relationships between distances.

Recent work [24] proposed using the mean average precision (MAP) as well. For a graph $G = (V, E)$, let $a \in V$ have neighborhood $N_a = b_1, b_2, ..., b_{\text{deg(a)}}$, where deg(a) denotes the degree of a. In the embedding f, consider the points closest to $f(a)$, and define R_{a,b_i} to be the smallest set of such points that contains b_i (that is, $R_{a,b_i}$is the smallest set of nearest points required to retrieve the ith neighbor of a in f). Then, the MAP is defined to be:

$$MAP(f) = \frac{1}{|V|} \sum_{a \in V} \frac{1}{\text{deg(a)}} \sum_{i=1}^{|N_a|} \frac{|R_{a,b_i} \cap N_a|}{|R_{a,b_i}|} \tag{5.10}$$

We have MAP(f) ≤ 1, with 1 as the best case. MAP is not concerned with explicit distances, but only ranks between the distances of immediate neighbors. It is a local metric.

Based on those 2 metrics we define two different loss functions. The first one, which we will call geometric loss 5.11 as it is based on all pairwise distances in the graph. By noting the graph as G and the manifold M, the loss id is defined by:

$$L(\Theta) = \sum_{1 \leq i \leq j \leq n} \left| \left(\frac{d_M(\theta_i, \theta_j)}{d_G(\theta_i, \theta_j)} \right)^2 - 1 \right| \tag{5.11}$$

The second one, which we will call contrastive loss 5.12 as it is based on sample of negative nodes, is not based on the graph structure directly but rather on hypernymy relation (a *IsAPartOf* b). In that case, embeddings are learned on the transitive closure of the graph (see 5.2b), the hierarchical structure is not directly visible from the raw data but has to be inferred. Let $T = \{(u, v)\}$ be the set of observed hypernyms relations between symbols and $N(u, v) = \{v' | (u, v') \notin T\} \cup \{v\}$ the set of negative sample for u (including v). The loss function is defined as:

$$L(\Theta) = \sum_{(u,v) \in T} \log \frac{\exp^{-d(u,v)}}{\sum_{v' \in N(u,v)} \exp^{-d(u,v')}} \tag{5.12}$$

We will analyse results for the two losses in this work. Geometric loss 5.11 is based on the actual distances in the graph and aims to preserve them, while contrastive loss 5.12 is based on hypernyms relations and just encourages related objects to be closer to each other than objects for which there is no relationship. As the number of negative nodes can be significant, we usually perform sampling here.

Besides for evaluation, we will also rely on the rank. For each observed relationship (u, v), we rank its distance $d(u, v)$ among the ground-truth negative examples for u, i.e., among the set $\{d(u, v) | (u, v) \notin T)\}$.

5.3 Experiments

Here, we intend to apply the optimisation process detailed in last section to the hierarchy of ICD and CPT codes.

We collected the codes of the two classifications by scraping websites. For each code, we collect both it's description and it's children. As a result, we are able to retrieve the global tree structure for each classification.

We encounter various difficulties in our implementation:

- The size of the classifications made the optimisation quite slow with our implementation. The model requires memory linearly proportional to the number of edges in the dataset. As a result we mainly iterated on a mock dataset.
- We struggled to reproduce the results displayed in [24]. In particular we found out that the process was heavily dependent on hyperparameters and prone to instabilities that are inherent to the Poincaré model for optimisation [38].
- We aimed to implement it from scratch as hyperbolic embeddings could be relevant for tasks different than automated medical coding at Qantev.

5.3.1 Technical implementation

Non-uniform negative sampling

While this is not precised in the original paper [24], the negative sampling of nodes is not uniform. Indeed, more common nodes should be more likely to be chosen and therefore far more updates are likely to be performed on these nodes, affecting the final vectors significantly.

Overfitting

We noticed that, for the contrastive loss almost all of the vectors were too close to the boundary. This makes sense mathematically, as Poincaré distances near the boundary change extremely fast, and therefore it is easier for the optimizer to achieve a lower loss. However, this is essentially a form of overfitting – our goal is to obtain useful representations, not simply achieve the minimum loss value. It seems, as suggested by the Gensim maintainers [39] that some regularization may be needed. A mathematical way to enforce this is by adding penalties on the norm of the parent node in an edge via L2 regularization. Note that L2 regularization is often used for all parameters, however here we're only using it for the parent node in a training example.

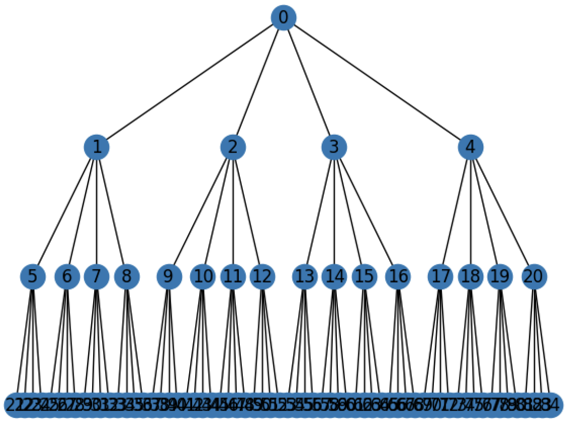
5.3.2 Results

Below, we present results for the various datasets we used. We rely on 2D embeddings that can be visually interpreted. When possible we also give distortion, rank and MAP; these metrics take a lot of time to be calculated (quadratic complexity with the number of nodes) and as such are only calculated at the end of the training. In all our experiments we used a batch size of 10, a sample

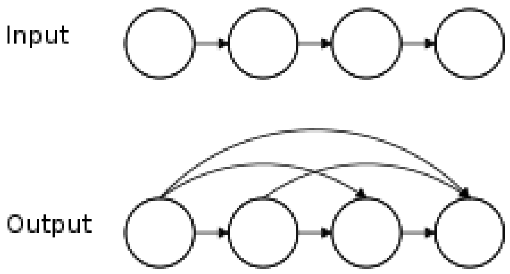
10 negatives nodes, 500 epochs, warm-up of 20 epochs (where the learning rate is 10 times smaller than the actual value) and a learning rate of 1 for contrastive loss and 3 for geometric loss.

Mock data

The mock data used corresponds to a tree with a top node, a branching factor of 4 and 4 different levels, see figure 5.2a.



(a) Mock data tree structure



(b) Transitive closure mechanism. Source: Wikipedia

Figure 5.2: Mock data used

The results obtained can be found on table 5.1 and 5.2, and the visualisations on figure 5.3.

Visually we can see that, as expected, the geometric loss preserves the graph geometry while the contrastive loss make sure to separate the graph in different clusters. This is also visible in the metrics displayed on the table 5.1 where the distortion for constrative loss is consistently higher. Besides, we can see that on constrastive loss the curvature has no to a small effect. This is expected as the contrastive loss doesn't rely on curvature but just on maximising distance between negative nodes.

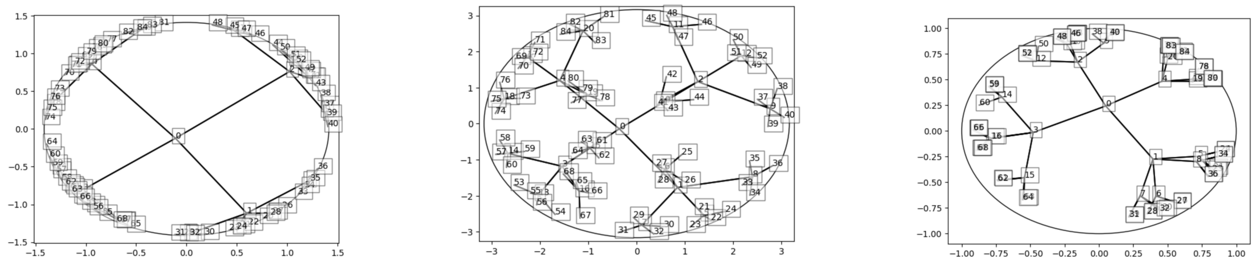
Interestingly, the geometric loss performs well on both local and global metrics. Besides, the curvature has a huge impact on it since this loss deals with global structure that is directly concerned by the space geometry. The amount of curvature is actually important enough that Gu et al. [40] tried to use curvature as a parameter to be optimized in the optimisation model. Let's note that we tried curvature lower than -1 but it resulted in many numerical instabilities. Additionally, a bigger size embedding does not always correlated with better metrics. For this mock data, the geometric loss seems to bring overall better results.

We display the results on mock data transitive closure on table 5.2 following the approach adopted in [24] where they only consider transitive closure results. We don't display the distortion in this table because the structure of the graph is perturbed by the numerous added edges through transitive closure mechanism. We can observe that all metrics on the transitive closure graph are somehow boosted compared to table 5.1. This is because each node gets more neighbors. Nonetheless, we consider that results on the mock data are more important as they deal directly with the underlying structure that we care about.

Let's note as well that the latency differs for each loss. For the contrastive loss, as we need to sample negative examples at each epoch, we actually train each epoch on a slightly different dataset. This sampling takes some times in our implementation that results in longer training (several hours/days depending on the number of nodes). On the contrary, we don't have such sampling with the geometric loss. Nonetheless, to use it we need to have distances between all

Models	**Curvature**	**Metrics**	**Dimension**		
			2	**10**	**50**
Contrastive loss	K=-1	Rank	3.18	2.72	2.65
		MAP	0.46	0.46	0.46
		Distortion	1.41	1.58	1.59
	K=-0.5	Rank	3.85	2.84	2.70
		MAP	0.43	0.52	0.53
		Distortion	2.06	2.33	2.38
	K=-0.1	Rank	4.42	2.59	2.44
		MAP	0.47	0.54	0.54
		Distortion	1.86	2.88	2.83
Geometric loss	K=-1	Rank	4.11	2.00	2.00
		MAP	0.30	0.55	0.56
		Distortion	0.12	0.02	0.02
	K=-0.5	Rank	2.08	**1.52**	2.04
		MAP	0.62	**0.96**	0.84
		Distortion	0.16	**0.03**	0.03
	K=-0.1	Rank	10.6	3.48	3.16
		MAP	0.15	0.34	0.35
		Distortion	0.23	0.05	0.04

Table 5.1: Embeddings results for mock data



(a) 2D Poincaré embeddings for mock data using contrastive loss

(b) 2D Poincaré embeddings for mock data using contrastive loss with a curvature K = -0.1

(c) 2D Poincaré embeddings for mock data using geometric loss

Figure 5.3: Various 2D visualisations of mock data embeddings

pairs (quadratic complexity with numbers of nodes). As the Floyd-Warshall algorithm to compute them is of cubic complexity, this takes lot of time once but only needs to be performed once.

Mammals dataset

The results we obtained on the mock datatset showed that hyperparameters have a huge impact on the embeddings (both visually and on metrics). Besides, the results seemed to be lower than what is presented in the literature. We wanted to benchmark our implementation so we test one of dataset used in the paper [24]. We used sub-category Mammals from WordNet [41]. Results are visible on table 5.3 and figure 5.4. We only tested the contrastive loss as this is the only one used in the paper [24].

Visually, the embeddings looks like expected. Nonetheless, the final metrics are lower than what was reported in the original paper. For comparison, on this specific subset [24] reported a MAP of 0.93 and a mean rank of 1.23 on 5 dimensions embeddings. Some other resources [39] also mentioned difficulties reproducing results.

Models	**Curvature**	**Metrics**	**Dimension**		
			2	**10**	**50**
Contrastive loss	K=-1	Rank	1.78	1.15	1.09
		MAP	0.85	0.97	0.97
	K=-0.5	Rank	3.49	2.06	1.87
		MAP	0.79	0.86	0.88
	K=-0.1	Rank	3.87	2.19	2.05
		MAP	0.81	0.87	0.87
Geometric loss	K=-1	Rank	5.7	3.3	3.3
		MAP	0.54	0.72	0.73
	K=-0.5	Rank	1.38	**1**	1.27
		MAP	0.96	**1**	0.98
	K=-0.1	Rank	4.63	1.90	1.72
		MAP	0.86	0.91	0.91

Table 5.2: Embeddings results for the mock data hypernyms

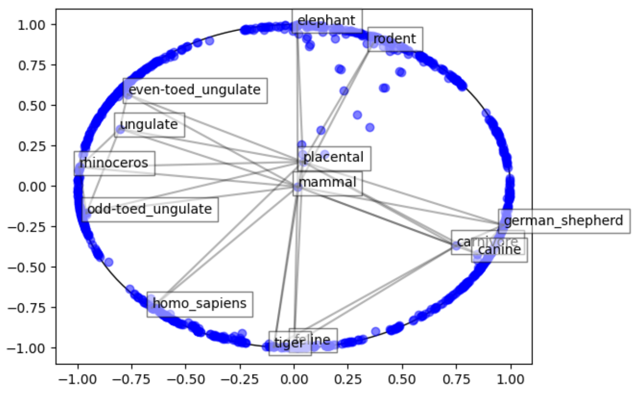


Figure 5.4: 2D Poincaré embeddings for Mammals

CPT

For automated medical coding, we care both about global and local metrics, so display results on all metrics.

Results are displayed on figure 5.5 and table 5.4, 5.5. We see that for CPT the MAP is at best 0.35, and it is 0.71 for CPT hypernyms. That means that on overage, for a particular code, there are roughly 65 % of his neighbors that shouldn't be so close to it in the embeddings space. We deemed that such results were not sufficient to use the hyperbolic embeddings in our model for automated medical classification presented in chapter 4. At the moment, the engineering required to incorporate it after the hierarchical attention module is likely to be too complex for an increase that is expected to be too low. Some engineering enhancements are required as well to improve the latency.

However, it is interesting to note that here contrastive loss seems to give better results than geometric loss. This may be explained by the fact that for trees with many nodes, geometric

Models	**Curvature**	**Metrics**	**Dimension**		
			2	**10**	**50**
Contrastive loss	K=-1	Rank	25.3	27.5	23.5
		MAP	0.38	0.69	0.71

Table 5.3: Embeddings results for Mammals hypernyms

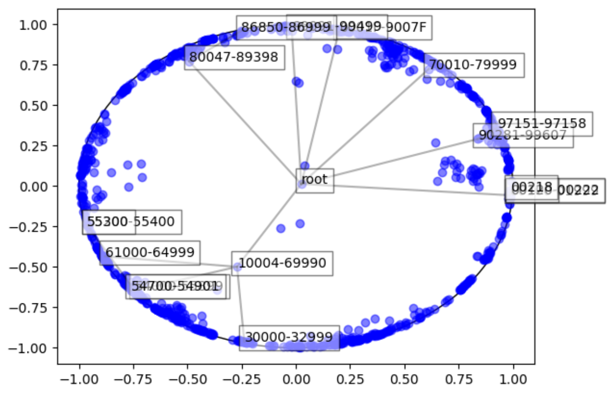


Figure 5.5: 2D Poincaré embeddings for CPT codes

Models	**Curvature**	**Metrics**	**Dimension**		
			2	**10**	**50**
		Rank	125	23.7	**12.9**
Contrastive loss	K=-1	MAP	0.07	0.33	**0.35**
		Distortion	0.35	0.39	**0.40**
		Rank	3950	3280	3245
Geometric loss	K=-1	MAP	1e-3	0.03	0.05
		Distortion	0.23	0.19	0.18

Table 5.4: Embeddings results for CPT

loss, which relies on all pairs, has difficulty optimizing them all at the same time. Indeed, in our experiments, we found that there is a part at the beginning of the training where the loss seems to be stable, but after a few hundred epochs, it suddenly decreases and then reaches another slowly decreasing state.

Besides it should be noted that the worse distortion obtained on contrastive loss here are order of magnitude lower than what was obtained in 5.1. Our interpretation is that this is simply a consequence of the number of nodes and the structure of the tree. Indeed, as the tree grows, the distance between points tends to be higher on average, but as the contrastive loss creates different groups of points that need to be further apart, the average distance between points is also high. For us, the fact that we get better distortion here is a purely statistical fact and should not be interpreted as an inherent property of contrastive loss.

ICD

The ICD classification is composed of 100,000 codes while the CPT classification is composed of 10,000 codes. Because of the latency encountered for CPT, we didn't launch any embeddings on ICD yet.

5.4 Potential use besides Automated Medical Coding

Our initial idea was to used the hyperbolic embeddings to guide automated medical coding, in the same way as Hypercore does [32]. This goal was not reached because of the high instability encountered with Poincaré model. The only off-the-shelf implementation that we found was the one of Gensim [42], but even in that case the results we obtained were highly unstable. Besides, it seems this part of Gensim is not really maintained as of today.

Models	**Curvature**	**Metrics**	**Dimension**		
			2	**10**	**50**
Contrastive loss	K=-1	Rank	317	18.8	**16.6**
		MAP	0.25	0.65	**0.71**
Geometric loss	K=-1	Rank	1425	320	282
		MAP	0.22	0.45	0.56

Table 5.5: Embeddings results for CPT hypernyms

Besides, the encouraging results showcased in [24] only concerns the transitive closure graph that is not of primary interest in our case. Still, as seen in section 4, inspired by the idea of hyperbolic embeddings we implemented a state-of-the-art hierarchical decoder.

Nevertheless, it is essential to resolve the obstacles we have encountered here, as hyperbolic embeddings could be relevant to different use cases in healthcare. An interesting framework would be that of fraud detection. Indeed, a standard method of fraud detection is code inconsistency on a claim. For example, an ICD indicating ”toothache” does not generally correspond to a CPT indicating ”pregnancy test”. The standard method for detecting this type of fraud is statistical testing of code co-occurrence. We could take advantage of the hyperbolic space by integrating ICD and CPT into the same space on the basis of their co-occurrence. The closer two codes are in hyperbolic space, the more likely they are to appear together in the same application. The further apart they are, the more likely they are to be inconsistent. Nevertheless, this method relies on integration in a space with non-constant curvature, which is studied in [40].

Conclusion and future work

In this work, we identify the main challenges of applying automatic medical coding, including long text input, large label set and mismatched domain, for both ICD and CPT classifications. We have integrated the previous state-of-the-art by adding a new hierarchical attention module to PLM-ICD. This new technique considerably improves results on rare medical codes.

In addition, we have shown that by incorporating a small amount of real data, it is possible to achieve better results on MIMIC-IV with minimal adaptation. The main bottleneck being the collection of additional data.

With regard to Poincaré embeddings, although we were unable to achieve our initial goal, we have highlighted the obstacles that still need to be overcome for them to become commonplace in deep learning (the main one being they are not, at the time of writing, part of any mainstream deep learning framework). In particular, we have shown that to critically assess the quality of embeddings without having downstream tasks, the full set of rank, MAP and distortion metrics is required to obtain the full embedding picture. In addition, 2D embeddings also provide a visual way of ensuring that embeddings retain hierarchy. Indeed, we are not directly interested in minimizing geometric or contrastive loss, but rather in obtaining robust embeddings for downstream tasks.

We hope that this work can pave the way for research into harnessing the great potential of automated medical coding. Research in this area is still active, and some future lines of work could include:

- Find new techniques to take into account the evolution of medical codes, from ICD-9 to ICD-11. To our knowledge, there is yet no literature on this subject.
- Find new techniques to enable the model to predict codes it has never seen in the training data. An interesting idea might be to have a tokenizer specifically for medical codes that aims to take into account the code hierarchy.
- Study different models of hyperbolic geometry (e.g. the Lorentz model) to reduce the instability encountered with the Poincaré model.

Bibliography

[1] J. Snow, “On the mode of communication of cholera,” *Edinburgh medical journal*, vol. 1, no. 7, p. 668, 1856.

[2] S. Ji, W. Sun, H. Dong, H. Wu, and P. Marttinen, “A unified review of deep learning for automated medical coding,” *arXiv preprint arXiv:2201.02797*, 2022.

[3] J. E. Harrison, S. Weber, R. Jakob, and C. G. Chute, “Icd-11: An international classification of diseases for the twenty-first century,” *BMC medical informatics and decision making*, vol. 21, no. 6, pp. 1–10, 2021.

[4] W. H. Organization and N. C. for Health Statistics (US), *The International Classification of Diseases, 9th Revision, Clinical Modification: Procedures: tabular list and alphabetic index.* Commission on Professional and Hospital Activities., 1980, vol. 3.

[5] S. Eslami, P. Adorjan, and C. Meinel, “Sehmic: Semi-hierarchical multi-label icd code classification.,” in *CLEF (Working Notes)*, 2020.

[6] E. Coiera, *Guide to Health Informatics (3rd ed.).* CRC Press, 2015. [Online]. Available: `https://doi.org/10.1201/b13617`.

[7] H. Dong, M. Falis, W. Whiteley, *et al.*, “Automated clinical coding: What, why, and where we are?” *NPJ digital medicine*, vol. 5, no. 1, p. 159, 2022.

[8] E. M. Burns, E. Rigby, R. Mamidanna, *et al.*, “Systematic review of discharge coding accuracy,” *Journal of public health*, vol. 34, no. 1, pp. 138–148, 2012.

[9] D. Lang, “Consultant report-natural language processing in the health care industry,” *Cincinnati Children’s Hospital Medical Center, Winter*, vol. 6, 2007.

[10] A. L. Goldberger, L. A. Amaral, L. Glass, *et al.*, “Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals,” *Circulation*, vol. 101, no. 23, e215–e220, 2000.

[11] K. Crawford, *The atlas of AI: Power, politics, and the planetary costs of artificial intelligence.* Yale University Press, 2021.

[12] A. E. W. Johnson, T. J. Pollard, L. Shen, *et al.*, “Mimic-iii, a freely accessible critical care database,” *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[13] A. Johnson, L. Bulgarelli, T. Pollard, S. Horng, L. Celi, and R. Mark, *Mimic-iv (version 1.0)*, 2020.

[14] J. Edin, A. Junge, J. D. Havtorn, *et al.*, “Automated Medical Coding on MIMIC-III and MIMIC-IV: A Critical Review and Replicability Study,” in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Taipei, Taiwan: ACM Press, 2023, ISBN: 978-1-4503-9408-6. DOI: `10.1145/3539618.3591918`.

[15] J. Mullenbach, S. Wiegreffe, J. Duke, J. Sun, and J. Eisenstein, “Explainable prediction of medical codes from clinical text,” *arXiv preprint arXiv:1802.05695*, 2018.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[18] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *Advances in neural information processing systems*, vol. 27, 2014.

[20] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[22] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[23] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[24] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," *Advances in neural information processing systems*, vol. 30, 2017.

[25] M. Gromov, "Hyperbolic groups," in *Essays in group theory*, Springer, 1987, pp. 75–263.

[26] N. Linial, E. London, and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications," *Combinatorica*, vol. 15, pp. 215–245, 1995.

[27] S. Bonnabel, "Stochastic gradient descent on riemannian manifolds," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2217–2229, 2013.

[28] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun, "Doctor ai: Predicting clinical events via recurrent neural networks," in *Machine learning for healthcare conference*, PMLR, 2016, pp. 301–318.

[29] T. Baumel, J. Nassour-Kassis, R. Cohen, M. Elhadad, and N. Elhadad, "Multi-label classification of patient notes a case study on icd code assignment," *arXiv preprint arXiv:1709.09587*, 2017.

[30] S.-C. Tsai, T.-Y. Chang, and Y.-N. Chen, "Leveraging hierarchical category knowledge for data-imbalanced multi-label diagnostic text understanding," in *Proceedings of the tenth international workshop on health text mining and information analysis (LOUHI 2019)*, 2019, pp. 39–43.

[31] T. Vu, D. Q. Nguyen, and A. Nguyen, "A label attention model for icd coding from clinical text," *arXiv preprint arXiv:2007.06351*, 2020.

[32] P. Cao, Y. Chen, K. Liu, J. Zhao, S. Liu, and W. Chong, "Hypercore: Hyperbolic and co-graph representation for automatic icd coding," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 3105–3114.

[33] C.-W. Huang, S.-C. Tsai, and Y.-N. Chen, "Plm-icd: Automatic icd coding with pretrained language models," *arXiv preprint arXiv:2207.05289*, 2022.

[34] Z. Zhang, J. Liu, and N. Razavian, "Bert-xml: Large scale automated icd coding using bert pretraining," *arXiv preprint arXiv:2006.03685*, 2020.

[35] J. Lee, W. Yoon, S. Kim, *et al.*, "Biobert: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[36] Y. Gu, R. Tinn, H. Cheng, *et al.*, "Domain-specific language model pretraining for biomedical natural language processing," *ACM Transactions on Computing for Healthcare (HEALTH)*, vol. 3, no. 1, pp. 1–23, 2021.

[37] P. Lewis, M. Ott, J. Du, and V. Stoyanov, "Pretrained language models for biomedical and clinical tasks: Understanding and extending the state-of-the-art," in *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, Online: Association for Computational Linguistics, Nov. 2020, pp. 146–157. DOI: `10.18653/v1/2020.clinicalnlp-1.17`. [Online]. Available: `https://aclanthology.org/2020.clinicalnlp-1.17`.

[38] F. Sala, C. De Sa, A. Gu, and C. Ré, "Representation tradeoffs for hyperbolic embeddings," in *International conference on machine learning*, PMLR, 2018, pp. 4460–4469.

[39] J. Jain, "Implementing poincarre embeddings," [Online]. Available: `https://rare-technologies.com/implementing-poincare-embeddings/`.

[40] A. Gu, F. Sala, B. Gunel, and C. Ré, "Learning mixed-curvature representations in product spaces," in *International conference on learning representations*, 2018.

[41] C. Fellbaum, *WordNet: An electronic lexical database*. MIT press, 1998.

[42] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," English, in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, `http://is.muni.cz/publication/884893/en`, Valletta, Malta: ELRA, May 2010, pp. 45–50.