CHIRon: A Generative Foundation Model for Structured Sequential Medical Data

Brian L. Hill
Optum AI

Melika Emami
Optum AI

Vijay S. Nori
Optum AI

Aldo Cordova-Palomera
Optum AI

Robert Tillman
Optum AI

Eran Halperin
Optum AI

Abstract

Recent advances in large language models (LLMs) have shown that foundation models (FMs) can learn highly complex representations of sequences that can be used for downstream generative and discriminative tasks such as text generation and classification. While most FMs focus on text, recent work has shown FMs can be learnt for sequential medical data, e.g. ICD-10 diagnosis codes associated with specific patient visits. These FMs demonstrate improved performance on downstream discriminative disease classification tasks, but cannot be used for generative tasks such as synthesizing artificial patient visits for data augmentation or privacy-preserving data sharing since they utilize BERT-based pre-training. In this paper, we introduce CHIRon, the first generative FM for sequential medical data. CHIRon utilizes causal masking during for pre-training, enabling generative applications, and incorporates a number of architectural improvements and support for additional medical data types (diagnoses, procedures, medications, lab results, place of service, demographics). We show empirically that CHIRon can be used to generate realistic sequential medical data and also outperforms state of the art FMs for sequential medical data on disease classification tasks.

1 Introduction

Foundation models (FMs) offer many improvements over traditional machine learning (ML) models, including better predictive performance, requiring less labeled data, and simplifying model deployment [24]. However, most prior work using FMs in the healthcare setting focuses on text such as clinical notes [8] or biomedical text [10], despite significant amounts of healthcare data such as administrative claims or electronic health records (EHRs) being stored in structured databases.

Several papers have developed FMs such as BERT [6] using structured sequential medical data [11,19], such as ICD-10 diagnosis codes associated with specific patient visits, and have shown promising improvements over traditional ML methods in downstream prediction tasks such as disease classification. These BERT-based FMs, however, cannot easily be used for generative purposes [16] – for example, generating synthetic visit sequences to enable privacy-preserving data sharing applications or augmenting existing patient data [26]. Given the recent success of generative FMs such as GPT-style models for text [3, 17, 18], we propose a novel generative FM for structured sequential patient data and investigate its performance on both generative and discriminative tasks.

In this work, we introduce CHIRon (Contextualized Healthcare Information RepresentatiON), the first generative FM trained on structured sequential medical data (rather than text). In addition to utilizing causal masking for pre-training the FM, we also introduce a number of architectural improvements and support for additional data types. While previous transformer-based models for sequential

1st Workshop on Deep Generative Models for Health at NeurIPS 2023.

medical data have focused specifically on diagnosis codes, we expand to include procedure codes, medications, lab results, and patient demographics. Additionally, we introduce a novel embedding for place of service information that adds useful context to each medical code. We show that CHIRon can accurately generate new codes using several quantitative metrics. Further, we fine-tuned CHIRon for disease onset classification and find that it outperforms existing state-of-the-art discriminative FMs for sequential medical data. Our experiments show that generative FMs can be powerful tools for both generating and classifying sequential medical data.

2 Related Work

Language model pretraining LLM pre-training has shown remarkable success in a variety of downstream tasks. These models efficiently use in-context information and eliminate the need for task-specific architectures. One of the most widely used models, BERT [6], is built on the Transformer [21] architecture and uses bidirectional context for learning representations. The core training objective employed by BERT is masked language modeling which encourages the model to better understand word relationships. GPT-style models [3, 17, 18] similarly uses a Transformer architecture but emphasize auto-regressive generation, which is useful in synthetic data generation. It scales up to billions of parameters and can perform both conditional and unconditional text generation. Like BERT, it can also be adapted for different NLP tasks by fine-tuning. Recent works have incorporated BERT and GPT for NLP tasks using medical text such as [2, 7, 8, 10, 14, 25].

Representation learning frameworks in the clinical domain One successful model that leveraged the temporal dependencies in clinical events is RETAIN [5], which used a two-level neural attention mechanism for learning visit representations. Papers such as BEHRT [11] and G-BERT [20] have attempted to employ contextualized pre-trained embeddings in the clinical domain. The former developed a model for diagnosis code prediction in different time windows and the latter leveraged graph neural networks for medication code prediction using a single-visit-level dataset. Perhaps the most relevant work to ours is Med-BERT [19]. The authors used a BERT model to learn contextualized diagnosis code embeddings along with a visit and/or positional embedding for downstream disease prediction tasks. While BERT models are able to effectively learn contextualized code representations, these models are not explicitly optimized for generation like GPT-style transformer models. In this paper, we adapt the framework of GPT-style models and pretrain our 6.3M parameter model on structured health records (rather than clinical text).

3 Methods

For our experiments we utilized Optum de-identified data which contains structured administrative claims and clinical data such as medical and pharmacy claims, lab results, demographics, and enrollment records for 44 million patients. The protocol and supporting materials representing this work were prospectively submitted to the UnitedHealth Group Office of Human Research Affairs for IRB review and were approved. We extracted demographics (age and sex), diagnosis codes, procedure codes, medications, and lab results, along with their corresponding encounter dates and place of service information, to build chronologically ordered lists of medical codes for each individual. See Appendix 6.1 for more details.

CHIRon pre-training CHIRon is a GPT-style [17, 18] model, where each medical code is represented as an individual token. We augment the GPT architecture with several additional embeddings to add healthcare-specific context to each code. In addition to the standard positional embeddings, we include visit embeddings (similar to [11, 19]), age embeddings (similar to [11]), and place of service embeddings, a novel data type. Place of service specifies where the service (code) took place, including these locations: outpatient, inpatient, emergency, custodial, independent lab, home (or unknown). This information adds important context – e.g., a diagnosis code for chest pain should have a different representation if it occurred in a primary care office versus an emergency room setting. Each of these embeddings are element-wise added to the code embeddings before being used as input to the model. We also prepend two tokens to every code sequence: one token indicating the sex of the patient and one token indicating the patient's age (binned into 5-year groups – e.g., 20-24, 25-29, etc.). The model was pre-trained using the causal language modeling objective as described in [18]. For additional model pre-training details see Appendix 6.3.

Sequential medical data generation To estimate the generative performance of CHIRon, we use a truncation procedure to remove medical codes from the end of a patient record, and evaluate how similar the generated codes are to the truncated codes. Specifically, we filter the pre-training validation set to select patients who have at least 50 codes. We truncate the last (most recent) T codes from each record, and these T codes are used as our reference (ground truth) code sequences. Using the truncated records as input to the model, the CHIRon model generates T additional codes for each record. We then compare the reference code sequences with the generated code sequences to determine model performance for this generative task.

Disease classification The pre-trained CHIRon model was then fine-tuned for five separate binary classification tasks: predicting disease onset for chronic kidney disease (CKD), chronic obstructive pulmonary disease (COPD), dementia, diabetes, and predicting CKD disease progression (CKD-P, from stage 1-3a to stage 3b+). Cohort creation details can be found in Appendix 6.1. Classification cohort sizes ranged from 382k (CKD-P) to 3.3M (diabetes) individuals (see Appendix Table 4). To fine-tune the model, we append a classification (CLS) token to the code sequence and add a feed-forward neural network layer on top of the final layer's classification token embedding. During fine-tuning, we allow the entire model to be updated.

For comparison, we used state-of-the-art and other common classification methods: gradient-boosted trees (GBT), RETAIN [4], and Med-BERT [19] (see Appendix 6.4 for baseline method details). While Med-BERT originally only used diagnosis codes, we included procedure codes and medications as input to the Med-BERT model for a more fair comparison, and denote this with "Med-BERT+" (see Appendix 6.5 for Med-BERT results using only diagnosis codes).

4 Experiments

Sequential medical data generation Just as generative models for text can be used to generate synthetic text sequences based on an initial prompt, we can similarly generate synthetic sequential medical data. For a given medical record, we can use the generative capabilities of the pre-trained CHIRon model to sample additional synthetic patient data.

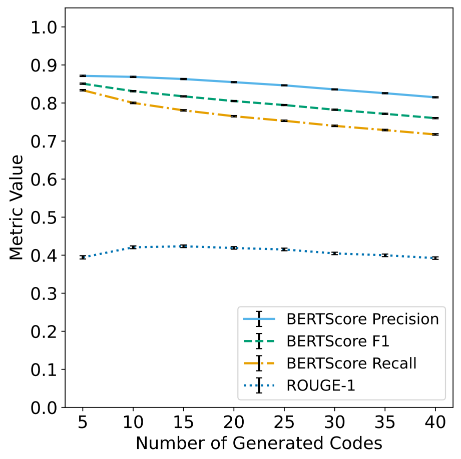


Figure 1: BERTScore and ROUGE metrics as a function of the number of codes truncated/generated. Error bars indicate bootstrapped 95% confidence intervals.

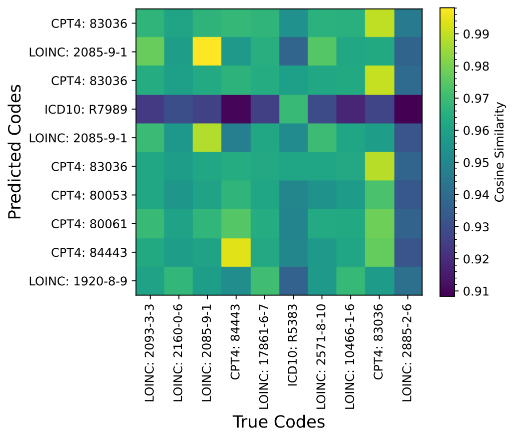


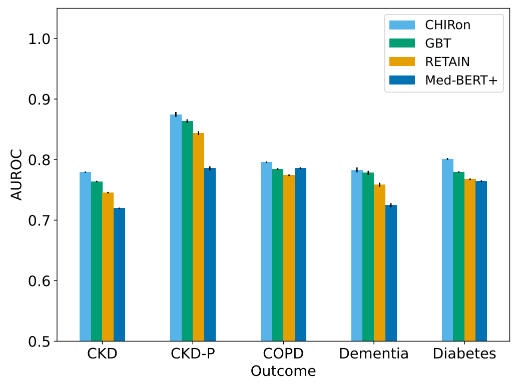
Figure 2: BERTScore cosine similarity heatmap comparing contextualized embeddings for $T = 10$ true (x-axis) and predicted (y-axis) codes from an example patient.

To quantitatively evaluate the generative performance, we adopt two established metrics from the NLP community: the ROUGE [12] score and the BERTScore [27]. The ROUGE-1 score measures the overlap of unigrams (single words/codes) between the reference sequence and the generated sequence. The BERTScore is a method for computing the similarity between two sequences as the mean cosine similarity between contextualized embeddings from the reference sequence and the generated sequences. Compared to the ROUGE score, the BERTScore penalizes a model less for generating codes that are very similar terms of medical taxonomy but not exact matches – as an example, if the model generates an ICD code "S92.812A" for a fracture of the left foot versus "S92.901A" for a fracture of the right foot. Using these two metrics allows us to quantify how well the model can generate medical codes both exactly and semantically.

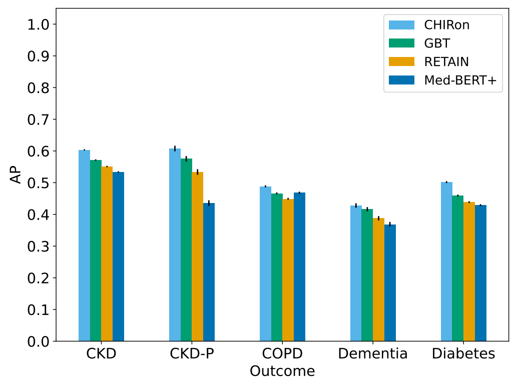
In Figure 1 we show ROUGE scores and BERTScore metrics as we vary the number of truncated/generated codes. Notably, the CHIRon code generation is more precise than it is sensitive. We find that the accuracy of the generated codes decreases as we truncate more codes from the record. This is expected – as more codes are truncated, more context is removed. In general, it is also more difficult to predict codes that occur farther in the future. Numeric performance metrics can be found in Appendix Table 5.

Figure 2 shows a BERTScore cosine similarity heatmap comparing contextualized embeddings from true and generated codes for an example patient. BERTScore precision searches for the highest similarity in each row, whereas BERTScore recall searches for the highest similarity in each column. As expected, the similarity is highest when the codes are an exact match. However, because the codes are contextualized, the same code at two different positions in the sequence can have different embeddings (and therefore different similarity to the query code).

Disease classification The pre-trained CHIRon model was fine-tuned for five binary classification tasks and compared with the baseline models. Figure 3 compares area under the ROC curve (AUROC) and average precision (AP) metrics for all models across the five tasks. In four out of five classification tasks, the GBT models were the strongest baseline, consistently outperforming both RETAIN and Med-BERT+ in terms of both AUROC and AP by a statistically significant difference. The fine-tuned CHIRon model achieved the highest AUROC and AP in four of the five classification tasks (CKD, CKD-P, COPD, diabetes) by a statistically significant margin, and did as well as the GBT model in the fifth task (dementia). See Appendix Tables 6, 7, 8, 9 and 10 for numeric results.



(a) Area under the ROC curve (AUROC)



(b) Average precision (AP)

Figure 3: Classification performance in terms of (a) area under the ROC curve (AUROC) and (b) average precision (AP, or area under the precision-recall (PR) curve) for each model across all disease outcomes. Error bars indicate bootstrapped 95% confidence intervals.

5 Discussion

In this work we developed CHIRon and showed that given sequential medical data the model is able to effectively generate realistic synthetic sequences of additional medical codes. Additionally, we found that fine-tuning this model for disease onset prediction achieves the best classification results in four of the five outcomes compared to three strong baseline methods. We note several limitations. We trained and validated the models on data from a single institution; in future work, we hope to validate model generalization to external datasets for both the generation and classification tasks. Additionally, due to the expensive nature of training large FMs, we were not able to conduct extensive hyperparameter tuning – results may improve with further investigation.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. *arXiv:1907.10902 [cs, stat]*, July 2019. arXiv: 1907.10902.

[2] Emily Alsentzer, John R Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew McDermott. Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323*, 2019.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners, July 2020. arXiv:2005.14165 [cs].

[4] Edward Choi, Mohammad Taha Bahadori, Joshua A. Kulas, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism, February 2017. arXiv:1608.05745 [cs].

[5] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. *Advances in neural information processing systems*, 29, 2016.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. arXiv: 1810.04805.

[7] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23, 2021.

[8] Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*, 2019.

[9] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 3149–3157, Red Hook, NY, USA, December 2017. Curran Associates Inc.

[10] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[11] Yikuan Li, Shishir Rao, José Roberto Ayala Solares, Abdelaali Hassaine, Rema Ramakrishnan, Dexter Canoy, Yajie Zhu, Kazem Rahimi, and Gholamreza Salimi-Khorshidi. BEHRT: Transformer for Electronic Health Records. *Scientific Reports*, 10(1):7155, April 2020. Number: 1 Publisher: Nature Publishing Group.

[12] Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[13] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019. arXiv:1711.05101 [cs, math].

[14] Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics*, 23(6):bbac409, 2022.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv:1912.01703 [cs, stat]*, December 2019. arXiv: 1912.01703.

[16] Ajay Patel, Bryan Li, Mohammad Sadegh Rasooli, Noah Constant, Colin Raffel, and Chris Callison-Burch. Bidirectional Language Models Are Also Few-shot Learners, February 2023. arXiv:2209.14500 [cs].

[17] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training.

[18] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners.

[19] Laila Rasmy, Yang Xiang, Ziqian Xie, Cui Tao, and Degui Zhi. Med-BERT: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction. *npj Digital Medicine*, 4(1):1–13, May 2021. Number: 1 Publisher: Nature Publishing Group.

[20] Junyuan Shang, Tengfei Ma, Cao Xiao, and Jimeng Sun. Pre-training of Graph Augmented Transformers for Medication Recommendation, November 2019. arXiv:1906.00346 [cs].

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[22] Paul J. Wallace, Nilay D. Shah, Taylor Dennen, Paul A. Bleicher, and William H. Crown. Optum Labs: Building A Novel Node In The Learning Health Care System. *Health Affairs*, 33(7):1187–1194, July 2014. Publisher: Health Affairs.

[23] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's Transformers: State-of-the-art Natural Language Processing, July 2020. arXiv:1910.03771 [cs].

[24] Michael Wornow, Yizhe Xu, Rahul Thapa, Birju Patel, Ethan Steinberg, Scott Fleming, Michael A. Pfeffer, Jason Fries, and Nigam H. Shah. The shaky foundations of large language models and foundation models for electronic health records. *npj Digital Medicine*, 6(1):1–10, July 2023. Number: 1 Publisher: Nature Publishing Group.

[25] Xi Yang, Aokun Chen, Nima PourNejatian, Hoo Chang Shin, Kaleb E. Smith, Christopher Parisien, Colin Compas, Cheryl Martin, Anthony B. Costa, Mona G. Flores, Ying Zhang, Tanja Magoc, Christopher A. Harle, Gloria Lipori, Duane A. Mitchell, William R. Hogan, Elizabeth A. Shenkman, Jiang Bian, and Yonghui Wu. A large language model for electronic health records. *npj Digital Medicine*, 5(1):1–9, December 2022. Number: 1 Publisher: Nature Publishing Group.

[26] Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. A survey of controllable text generation using transformer-based pre-trained language models. *CoRR*, abs/2201.05337, 2022.

[27] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation with BERT, February 2020. arXiv:1904.09675 [cs].

6 Appendix

6.1 Datasets and Cohort Selection

Datasets: The Optum de-identified dataset consists of de-identified administrative claims data such as medical and pharmacy claims, laboratory (labs) results, enrollment records, and demographic (age and sex only) information. Our cohorts included individuals with records occurring between 1 January 2014 and 31 December 2018. Individuals younger than 18 years or older than 89 years were excluded from our analysis. For all individuals we extracted a list of all medical codes with their corresponding timestamp and place-of-service information, and ordered the codes chronologically.

Diagnoses were coded using the International Classification of Diseases (ICD), Ninth and Tenth Revisions, Clinical Modification (ICD-9-CM and ICD-10-CM). Any ICD-9 codes were mapped ICD-10. Procedures were coded using the Current Procedural Terminology (CPT-4). For lab results, we use the Logical Observation Identifier Names and Codes (LOINC) standardized system to identify specific lab tests. Medications were grouped by generic name. American Medical Association (AMA) place-of-service code information for each medical code was categorized as one of these six categories: outpatient, inpatient, emergency, custodial, independent lab, home, or otherwise unknown.

The structured healthcare records were stored in a table with the following information:

- id: unique patient identifier
- enc_dt: date of encounter
- code_type: type of code - e.g., diagnosis (DIAG_ICD10), procedure (PROC_CPT4), etc.
- code: diagnosis (e.g., N18.1), procedure (e.g., 99214), etc.
- pos: place of service code

Pre-training cohort: We combined both the claims and clinical datasets into a single large dataset to pre-train the transformer-based models. We required that individuals be enrolled for 3 or more months, and have 6 or more days with encounters (diagnosis, procedure, lab, medication fill record) to be included. Identical records from claims and clinical data datasets were de-duplicated. This combined dataset included over 13.88 billion medical codes from 44,169,102 unique individuals. See Table 1 for counts of individuals, total number of ICD codes, and unique number of ICD codes for each dataset in our pre-training cohort.

Table 1: For each dataset we report the number of unique individuals, number of total ICD codes, number of unique ICD codes, the median (IQR) number of codes per each individual, the median (IQR) number of encounters per each individual, the median (IQR) age in years, and the percent female.

Dataset	Individuals	Total records	Unique codes	Codes per patient	Encounters per patient	Age (years)	Sex (%F)
Claims	21,493,605	6.55B	7,615	164 (72-386)	35 (15-81)	51 (35-64)	57.3
EHR	25,669,520	7.52B	7,261	151 (69-337)	19 (9-42)	50 (34-63)	59.8
Combined	44,169,102	13.88B	7,922	165 (73-378)	26 (12-61)	51 (34-64)	58.5

Fine-tuning cohorts: For the classification cohorts used for fine-tuning the CHIRon model and method comparison, we again leveraged the combined claims and clinical datasets. In the classification, we required that individuals be enrolled for a minimum of two years to ensure a minimum time window for outcomes to occur, and have 6 or more days with encounters.

Cases (positive labels) were defined as individuals who had at least two occurrences of any inclusion code (see Table 2) between 7-365 days apart in any setting (inpatient, outpatient, etc.), or one occurrence of a diagnosis code (see Table 2) in an inpatient setting (inpatient setting is more reliable). We used the earliest occurrence of any inclusion code as the "index date".

We excluded individuals if they met our exclusionary criteria: for CKD/CKD-P, if individuals had a diagnosis code for acute and unspecified renal failure, other specified and unspecified diseases of

Table 2: Medical codes used to define classification cohort cases. Case definition required at occurrence of at least two of these codes within a 365-day period. For lab values, we report the LOINC code and the lab result range that would count as evidence of the outcome (e.g., a LOINC 4548-4 ("Hemoglobin A1c/Hemoglobin.total in Blood") result between 6.501% and 100% is evidence of diabetes). ICD: International Classification of Diseases; CCSR: Clinical Classifications Software Refined; ETG: Episode Treatment Group; CPT-4: Current Procedural Terminology; PCC: ; LOINC: Logical Observation Identifier Names and Codes.

Outcome	ICD	CCSR	ETG	CPT-4	PCC	LOINC (min, max) lab range
CKD	E11.21, E11.29, N08	GEN003				33914-3 (0,75), 62238-1 (0,75), 98979-8 (0,75)
CKD-P	N18.32, N18.3, N18.5, N18.6					33914-3 (0,45), 62238-1 (0,45), 98979-8 (0,45)
COPD		RSP008			574, 576, 577, 605	
Dementia			239000, 316400	99307, 99308, 99309, 99310	344	
Diabetes		END003, END005			500-504, 513-515, 518, 520, 523, 841	1558-6 (126,1200), 2339-0 (150,1200), 2345-7 (150,1200), 27353-2 (150,1200), 4548-4 (6.501,100)

kidney and ureters, or kidney transplant status; for dementia, if individuals had a procedure code indicating they were living in a skilled nursing facility or hospice; for diabetes if an individual had any diagnosis codes that indicated Type-1 diabetes (see Table 3 for codes used).

We defined controls (negative labels) as individuals who had neither an inclusion code nor an exclusion code over the entire period for which we have their clinical history. Controls were randomly selected so that the final cohort had a case:control ratio of 1:5 (i.e., a 16.67% prevalence). Final cohort counts can be found in Table 4.

The cohorts were split into train and test using a time-based split, such that individuals with the most recent 20% were included in the test set and the remaining (earliest) 80% were used for model training. The training set was further randomly split into training and validation sets, such that 70% of the total data was used for model training and the remaining 10% was used for validation.

6.2 Preprocessing

Lab Results: Previous methods likely ignored lab results because they are noisy and need to be converted into discrete tokens. However, lab results contain specific and objective information about patient state, and can be used to more accurately phenotype patients for disease labeling. To incorporate the labs into our modeling framework, we used the following tokenization procedure:

1. Select LOINC codes with at least 1000 observations.
2. Compute deciles for each LOINC code.
3. Drop LOINC codes where the max value for third decile is 0, to remove labs where a significant number of results were zero-filled by an upstream data management process as the result data was unavailable.

Table 3: Medical codes used to exclude individuals from our classification cohorts. Any occurrence of these codes would exclude an individual from being a case or control for each outcome. ICD: International Classification of Diseases; CCSR: Clinical Classifications Software Refined; ETG: Episode Treatment Group; CPT-4: Current Procedural Terminology; PCC: ; LOINC: Logical Observation Identifier Names and Codes.

Outcome	ICD	CCSR	ETG	CPT-4	PCC	LOINC (min, max) lab range
CKD	Z94.0	GEN002, GEN006				
CKD-P	Z94.0	GEN002, GEN006				
COPD						
Dementia				99307, 99308, 99309, 99310		
Diabetes		END004				

Table 4: Classification cohort sizes and case counts. Using our case/control sampling procedure, each cohort maintains a 1:5 case:control ratio (i.e., a prevalence of 16.67%).

Outcome	Individuals	Number of cases
CKD	2,427,678	404,613
CKD-P	382,632	63,772
COPD	3,197,016	532,836
Dementia	583,896	97,316
Diabetes	3,299,352	549,892

4. Fit an exponential function using the maximum values from first 9 deciles. Use this function to compute the 10th decile.
5. Drop observations (tokens) which are more than $3\times$ the predicted 10th decile (from Step 4).

Therefore, each lab result token used as input to our model denotes the decile of the lab result for that specific test. For example, for a "Hemoglobin A1c/Hemoglobin. total in Blood" lab result (LOINC 4548-4) that fell into the 7th decile of the population distribution, we would denote the lab result token as "LABS_LOINC_4548-4-7" where the decile is added as a suffix to the token ID.

Tokenization: We built a tokenizer similar to [19], where each token corresponds to a unique medical code, using the Combined Dataset. Rather than use a vocabulary that included all 60,817 unique medical codes in the combined dataset, we selected codes that occurred in at least 1/1000 individuals, given that many codes are rare and only occur in a small subset of patients. This prevalence-based filtering left us with a tokenizer vocabulary of 7,922 unique medical codes. Any medical code that did not pass this prevalence-based threshold was not discarded but instead renamed as a "rare code" that was specific to the type of medical code (e.g., "DIAG_ICD10_RARE_CODE" or "PROC_CPT4_RARE_CODE").

6.3 Model Development

CHIRon is a GPT-based model and we adopt a similar architecture and pre-training techniques as GPT-2 and build on top of them.

Input Representation: A patient record consists of a series of encounters (i.e. visits), each containing several medical code tokens including diagnosis, procedure, and medication codes as well as tokenized lab results as explained above. Let $\mathbf{x}_c = (c_{11}, \cdots, c_{1n_1}, \cdots, c_{K1}, \cdots, c_{Kn_K})$ be the code sequence for patient X with K total visits where c_{ij} corresponds to the j-th code (token) that occurred in visit i, $i \in [K], j \in [n_i]$ where n_i is the total number of codes for visit i. For each encounter, the information about its place of service (one of a total of 7 categories), its timestamp which represents age of the patient in months at the time of encounter, and the visit number in clinical history is available as well. Given the total code sequence has N codes, let the following denote the context information sequences:

$$\begin{aligned}
\text{place of service: } \mathbf{x}_s &= (\underbrace{s_1, \cdots, s_1}_{n_1 \text{ count}}, \cdots, \underbrace{s_K, \cdots, s_K}_{n_K \text{ count}}), s_i \in \{0, 1, \cdots, 6\} \\
\text{visit number: } \mathbf{x}_v &= (\underbrace{1, \cdots, 1}_{n_1 \text{ count}}, \cdots, \underbrace{K, \cdots, K}_{n_K \text{ count}}), \\
\text{age: } \mathbf{x}_a &= (a_1, \cdots, a_N), \\
\text{position: } \mathbf{x}_p &= (1, \cdots, N),
\end{aligned}$$

where $\mathbf{x}_p$ corresponds to the position of codes in the sequence. Therefore, we represent a patient $X = \{\mathbf{x}_c, \mathbf{x}_s, \mathbf{x}_v, \mathbf{x}_a, \mathbf{x}_p\}$ as a collection of these sequences describing the medical record. Note that the record is organized chronologically with random ordering of the codes inside a visit. The tokenized code sequence is prepended with demographic tokens including a token c_a for the patient age (in years, binned into 5-year age groups – e.g., 20-24, 25-29, etc.) of the patient and a token c_s for patient sex (male or female): $\mathbf{x}_c = (c_a, c_s, c_{11}, \ldots, c_{Kn_K})$. Other context information sequences are also padded at the beginning accordingly. We utilize five different embedding layers to construct the final input sequence to the transformer model: (i) code embeddings $W_c \in \mathbb{R}^{|\text{vocab}| \times m}$, (ii) visit embeddings $W_v \in \mathbb{R}^{\text{max visit size} \times m}$, (iii) place-of-service embeddings $W_s \in \mathbb{R}^{|pos| \times m}$, (iv) time/age embeddings $W_a \in \mathbb{R}^{\text{max age} \times m}$ and finally, (v) standard positional embeddings $W_p \in \mathbb{R}^{\text{max seq length} \times m}$, where m is the embedding size. Each element of the padded $\mathbf{x}_c, \mathbf{x}_s, \mathbf{x}_v, \mathbf{x}_a$, and $\mathbf{x}_p$ sequences are then one-hot encoded to the desired dimensions |vocab|, |pos|, max visit size, max age, max seq length, and passed through the embedding layers. The output of the embedding layers are then added up together to construct the input to the CHIRon transformer model.

Architecture and Hyperparameters for Pre-Training (PT): We implemented the CHIRon architecture using the HuggingFace transformers [23] package (v.4.25.1) and Pytorch [15] (v2.0.1). The model contains a total of 6,392,832 parameters. For the transformer architecture of the CHIRon we used 6 layers, 8 heads, and embedding dimensionality of 256. The maximum sequence length is set to 512 and the inner feed forward layers have a dimension of 512. We also used the default attention dropout ratio and initializer range. We used the AdamWeight decay optimizer [13] with coefficient 0.01 and trained the model for 5e6 steps with early stopping of patience 3 using 2 Nvidia Tesla V100 GPUs.

Architecture and Hyperparameters for Fine-Tuning (FT): In fine-tuning, the code sequence is appended with a [CLS] token to use for classification. Our disease onset classification tasks are binary classification and we put a logistic FFL prediction head on top of the final layer of CHIRon. The fine-tuning transformer architecture is similar to the pretrain model. Starting from the pre-trained model, we train a separate model for each condition for 20 epochs with early stopping of patience 3 and batch size 64 using a single Nvidia Tesla V100 GPU.

Generation Procedure: We used the transformers *.generate()* function for auto-regressive generation of new codes. We used greedy search for generation and suppressed the rare codes "[CODE_TYPE_RARE_CODE]". The generation of new codes takes place one code at a time, i.e. the generated code at time t is used in the sequence for the generation of the code at time $t + 1$. We also make use of the other additional context information in the generation process and pad them at each time step: place of service is padded with the unknown token and the other sequences such as visit number and age of encounter (in months) are padded with their most recent value.

6.4 Model Comparisons

6.4.1 Gradient-boosted trees (GBT)

Gradient boosted trees (GBTs) which modeled the presence of codes, and not the sequence in which they occurred, were trained as one of the baseline models. Each cell in the model matrix was populated with the number of days a patient had a specific code. The demographic information was added using binary columns. A threshold of 0.0001 was set for initial feature selection – i.e., a code should appear in at least 10,000 patients to be included in the initial feature set. A two step approach was implemented for tuning the models. In the first step 150 Optuna trials were run and the features which had at least 1 split in 20% of those trials were selected for the second step. In the second step 300 Optuna trials were run on the pruned set of features. Average precision was the measure selected for early stopping the boosted tree training runs and Optuna trial evaluation. Libraries used for this comparison included lightgbm [9] (v3.3.5) and Optuna [1] (v3.1.0) for hyperparameter tuning.

6.4.2 RETAIN

RETAIN [5] is an interpretable two-level neural attention predictive model with applications to EHR data. It considers both the visit-level and the variable-level influence of each visit through two sets of attentions weight. It generates the attention vectors by running two RNNs backward in time and then creates the final context vector to use for classification. For more details on the model architecture see [5]. We used a Pytorch-lightning implementation of the original RETAIN code[1] based on modifications done in this repo [2]. We included the place of service, visit number, and time/age at encounter as well as demographic information as additional features in the model. For each condition, we trained a separate model for each of the three age brackets and data sources and aggregated the results in Figure 3. We did a coarse grid search hyperparameter tuning of the diabetes model and used the parameters for the rest of the RETAIN models.

6.4.3 Med-BERT

Med-BERT [19] is a BERT-based model that, instead of using text as input, learns contextualized representations of sequentially-ordered medical codes. Similar to BERT [6] models, Med-BERT can be pre-trained using an objective like masked language modeling on a large, general cohort of patients, and then fine-tuned on a smaller, more specific cohort for downstream tasks, e.g., disease onset prediction.

We implemented the Med-BERT architecture using the HuggingFace transformers [23] package (v.4.25.1) and Pytorch [15] (v2.0.1) using the same hyperparameters as described in [19]. We then pre-trained a Med-BERT model from scratch using the same preprocessing procedure such as not using the [CLS] token for input representation as described in [19]. We used the same tokenizer as the CHIRon model (described above in 6.2).

There are a few differences between the original Med-BERT [19] training and the Med-BERT training results presented in this paper: (1) We only pre-train the Med-BERT on the masked language modeling task. The original Med-BERT paper also includes a "prediction of prolonged length of stay in hospital" task. To perform this, the authors only include *inpatient* admissions in their data. However, we consider *all* types available encounters. (2) The original Med-BERT paper uses only diagnosis codes. To provide a fair comparison between this model and our CHIRon model, we trained two sets of Med-BERT models, one using only the diagnosis codes (ICD-10) in our data (denoted "Med-BERT"), and one with also including the procedure and medication codes (denoted "Med-BERT+"). Comparisons of the two models is included in Figure 4 and Tables 6,7,8,9, and 10.

To fine-tune the Med-BERT models, we attached a classification head to the pre-trained Med-BERT model using the "Med-BERT_only (FFL)" head, in which a feed-forward layer (FFL) is added on top of the first token's embedding as in the original Med-BERT implementation [3]. We fine-tuned the model using a batch size of 64 for a maximum of 20 epochs, with early stopping if the model's validation loss does not decrease after 3 epochs.

The pre-training and fine-tuning cohorts that were used are the same as the cohorts used for CHIRon.

6.5 Evaluation

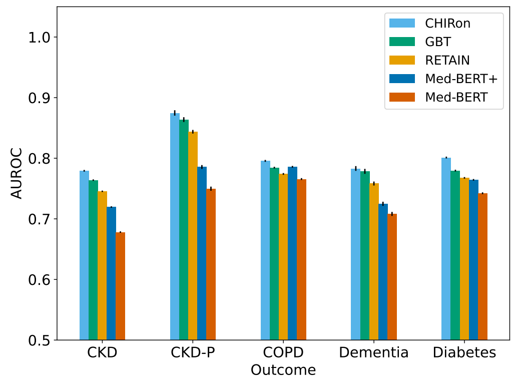
[1] `https://github.com/mp2893/retain`

[2] `https://github.com/ast0414/pytorch-retain`

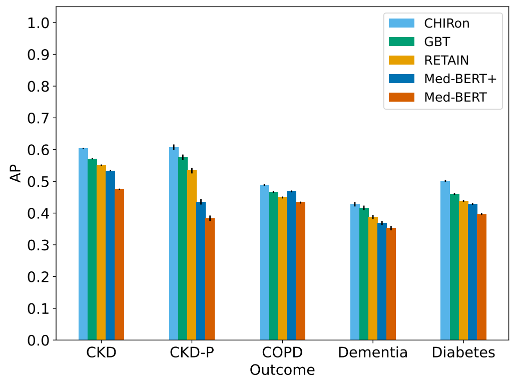
[3] `https://github.com/ZhiGroup/Med-BERT`

Table 5: Generative performance metrics as a function of the number of codes (NC) generated. The BERTScore F1 metric is a measure of cosine similarity between the generated medical codes and the true medical codes. The ROUGE-1 score measures the overlap of code unigrams (single codes) between the generated medical codes and the true medical codes. Error bars indicate bootstrapped 95% confidence intervals.

	BERTScore F1	BERTScore Precision	BERTScore Recall	ROUGE-1
NC				
5	0.851 (0.849-0.853)	0.871 (0.870-0.873)	0.834 (0.832-0.835)	0.394 (0.390-0.399)
10	0.831 (0.829-0.833)	0.869 (0.867-0.870)	0.800 (0.798-0.802)	0.421 (0.417-0.424)
15	0.818 (0.816-0.819)	0.863 (0.861-0.865)	0.781 (0.779-0.783)	0.423 (0.420-0.427)
20	0.805 (0.804-0.807)	0.855 (0.853-0.856)	0.765 (0.763-0.767)	0.419 (0.416-0.423)
25	0.795 (0.793-0.796)	0.846 (0.845-0.848)	0.753 (0.751-0.755)	0.415 (0.412-0.419)
30	0.782 (0.781-0.784)	0.836 (0.834-0.837)	0.740 (0.738-0.742)	0.405 (0.401-0.408)
35	0.772 (0.770-0.773)	0.826 (0.824-0.827)	0.729 (0.727-0.731)	0.400 (0.396-0.403)
40	0.760 (0.759-0.762)	0.815 (0.814-0.816)	0.717 (0.715-0.719)	0.392 (0.389-0.396)



(a) Area under the ROC curve (AUROC)



(b) Average precision (AP)

Figure 4: Classification performance in terms of (a) area under the ROC curve (AUROC) and (b) average precision (AP, or area under the precision-recall (PR) curve) for each model across all disease outcomes. The Med-BERT model uses only diagnosis codes, as in the original paper, and Med-BERT+ uses diagnosis codes, procedure codes, and medications. Error bars indicate bootstrapped 95% confidence intervals.

Table 6: Classification performance metrics for CKD

metric	AUROC	AP
model		
CHIRon	0.779 (0.778-0.780)	0.603 (0.601-0.606)
GBT	0.764 (0.763-0.765)	0.571 (0.569-0.574)
RETAIN	0.745 (0.744-0.747)	0.551 (0.549-0.553)
Med-BERT+	0.720 (0.718-0.721)	0.534 (0.532-0.536)
Med-BERT	0.678 (0.677-0.680)	0.475 (0.473-0.477)

Table 7: Classification performance metrics for CKD-P

metric	AUROC	AP
model		
CHIRon	0.875 (0.872-0.878)	0.608 (0.599-0.616)
GBT	0.864 (0.861-0.867)	0.576 (0.567-0.584)
RETAIN	0.844 (0.840-0.847)	0.534 (0.525-0.543)
Med-BERT+	0.786 (0.782-0.790)	0.436 (0.427-0.445)
Med-BERT	0.750 (0.745-0.754)	0.384 (0.375-0.392)

Table 8: Classification performance metrics for COPD

metric model	AUROC	AP
CHIRon	0.796 (0.794-0.797)	0.488 (0.485-0.491)
GBT	0.785 (0.783-0.786)	0.466 (0.463-0.469)
RETAIN	0.774 (0.773-0.776)	0.449 (0.446-0.452)
Med-BERT+	0.786 (0.785-0.787)	0.468 (0.465-0.471)
Med-BERT	0.766 (0.764-0.767)	0.433 (0.430-0.436)

Table 9: Classification performance metrics for Dementia

metric model	AUROC	AP
CHIRon	0.783 (0.780-0.786)	0.428 (0.421-0.435)
GBT	0.778 (0.775-0.782)	0.416 (0.410-0.423)
RETAIN	0.758 (0.755-0.762)	0.388 (0.381-0.395)
Med-BERT+	0.725 (0.721-0.729)	0.369 (0.362-0.376)
Med-BERT	0.708 (0.704-0.712)	0.353 (0.346-0.360)

Table 10: Classification performance metrics for Diabetes

metric model	AUROC	AP
CHIRon	0.801 (0.800-0.802)	0.502 (0.499-0.505)
GBT	0.780 (0.778-0.781)	0.459 (0.456-0.462)
RETAIN	0.768 (0.766-0.769)	0.439 (0.435-0.441)
Med-BERT+	0.764 (0.763-0.766)	0.429 (0.426-0.432)
Med-BERT	0.742 (0.741-0.744)	0.396 (0.394-0.399)