

©Copyright 2024

Mike A. Merrill

Multimodal Models for Time Series and Text

Mike A. Merrill

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2024

Reading Committee:

Tim Althoff, Chair

Shwetak Patel

Luke Zettlemoyer

Program Authorized to Offer Degree:

Computer Science and Engineering

University of Washington

Abstract

Multimodal Models for Time Series and Text

Mike A. Merrill

Chair of the Supervisory Committee:

Tim Althoff

Computer Science and Engineering

Time series are critical data which drive countless decisions in finance, healthcare, and science. However, multimodal NLP research has mostly focused on images and video. Here I enumerate barriers towards such models and describe my work towards mitigating them. I detail new multimodal NLP tasks for reasoning about time series, describe an LLM-powered agent that can answer questions about time series, and present methods for pretraining time series encoders. I also share work on using language models for code generation to assist scientists.

ACKNOWLEDGMENTS

To **Galen Weld**, my alpine mentor and labmate, for inspiring my love of the Cascades and making me feel so at home at UW and in Seattle. I'll miss you busting through the door of our lab, smiling wide, phone in hand to show me a video of your latest adventure.

To **Nick Randazzo** and **Joe Breda** for being such close friends in a difficult time in my life. I'll always cherish our weekend coffees, impulsive purchases, and enrichment activities.

To **Tanzeem Choudhury**, **Hane Aung**, and **Amy Sindone** for taking a chance on me back at Cornell and setting all of this in motion. I owe my career to you. And to the other members of the PAC Lab for giving a precocious undergrad a place to sit.

To **Anna Kornick** and her family for their generosity, hospitality, and support through the hardest years of this program.

To **Ted** and **Allison Benge**, **Mike McCool**, **Rachel** and **Ricky**, **John T.**, **Nick W.**, **Wilson Parry**, **Nikhil**, and **OSAT** for reminding me what's important.

To **Gabe Dreiman** for challenging my thinking while simultaneously being my biggest echo chamber.

To my friends from before **Nicole Hamilton**, **Tyler Neace**, **Javi Ortiz**, **Dan Masetti**, **Mauricio Moreyra**, **Felix Chen**, **Michelle Digiglio**, **Jason Freedman**.

To **Alana Zendano** for reminding me to lay down.

To my academic siblings **Advait Bhat**, **Deniz Nazarova**, **Mickel Liu**, **Inna Lin**, **Ken Gu**, **Ashish Sharma**, **Margaret Li**, **Xinyi Zhou**, **Jina Suh**, **Bret Nestor**, and **Vinayak**

Gupta for the years of enriching discussions over Diet Cokes. And to other academic friends: **Ari Holtzman, Ben Kushigian, Ludwig Schmidt** and the **EFML Lab, Josh Gardner, Dan Sturm, Girish Narayanswamy, Luca Foschini, Tom Hartvigsen, Mingtian Tan, Race Condition Running**, and the **academic advising staff at UW CSE**.

To **Tim Althoff**. It's trite, but by far the most important decision you'll make in grad school is picking your advisor. I'm so fortunate to have picked you.

To my Google collaborators, **Xin Liu, Daniel McDuff**, and **Shwetak Patel** for giving me space to work on PHIA and develop it into something I'm so proud of.

To the other members of my committee **Martin Saveski** and **Luke Zettlemyer** for their feedback.

To **Chris Rytting** for pushing me to be the best version of myself. There's no one I'd rather have tell me that I'm being lazy or thinking wrong. It's unfortunate how often you're right!

To my adopted families, the **Thomases, Wszaleks, Vokeses**, and **Chens**. My family never felt small because you made it so full.

To **Theresa Kennedy**, my godmother, for thirty years of spiritual guidance.

To every teacher who gave me a second chance to turn in an assignment, apologize, or sit down. Had I been born into other circumstances I would have slipped through the cracks long ago. I am here only because of the relentless patience of a few kind adults who saw something in me.

To my sister **Maeanna** and her wife **Rebecca**. Maeanna, I remember sitting in a rented RV with you and Pam Chen in Myrtle Beach around 2004. We were horrified when she told us she was best friends with her brother. But sure enough, twenty years later, here we are. I love you.

And, of course, to my parents **Mike D.** and **Melanie Merrill** for everything else. Every day I discover more of your love for me.

DEDICATION

To my grandparents, four teachers:
George Graban, Audrey Graban, Gail Harkness, and Jean Steel.

Here there is something we call elucidation

Is it the truth or merely a description?

DAVID BYRNE

TABLE OF CONTENTS

	Page
Chapter 1: Introduction	1
Chapter 2: Language Models Still Struggle to Zero-shot Reason about Time Series	5
2.1 Related Work	7
2.2 Forms of Time Series Reasoning	8
2.3 Dataset	10
2.4 Etiological Reasoning: Near Random Performance	13
2.5 Question Answering: Trailing Behind Human-Level Proficiency	15
2.6 Context-Aided Forecasting	19
2.7 Limitations	20
2.8 Conclusion	21
Chapter 3: Answering Open Ended Personal Health Questions using Large Language Model Agents	22
3.1 Related Work	25
3.2 Personal Health Insights	26
3.3 PHIA: The Personal Health Insights Agent	30
3.4 Evaluating Responses to Queries	33
3.5 Discussion	44
3.6 Limitations and Future Work	47
3.7 Conclusion	48
Chapter 4: Self-Supervised Pretraining and Transfer Learning Enable Flu and COVID-19 Predictions in Small Mobile Sensing Datasets	50
4.1 Related Work	54
4.2 Dataset	55

4.3	Challenges in Evaluation	58
4.4	Model Architecture	62
4.5	Model Performance	63
4.6	Comparing Self-Supervised Pretraining Tasks	64
4.7	Conclusion	74
Chapter 5:	MULTIVERSE: Mining Collective Data Science Knowledge from Code on the Web to Suggest Alternative Analysis Approaches	76
5.1	Related Work	79
5.2	Multiverse Analysis Tasks	81
5.3	Key Ideas	83
5.4	Methods	88
5.5	Evaluation	93
5.6	Error Analysis	101
5.7	Conclusion	102
Chapter 6:	Conclusion	103
Appendix A:	PHIA - The Personal Health Insights Agent	135
A.1	Additional Results	135
A.2	Additional Examples of Agent Behavior	136
A.3	Examples of Annotator Responses	140
A.4	Objective Personal Health Queries	141
A.5	Open-Ended Personal Health Insights Queries	143
A.6	Synthetic Wearable Users	150

Chapter 1

INTRODUCTION

Time series data are easily ignored yet critically important to the day to day function of finance, healthcare, and science. Daily revenue for a small business, your heart rate during a run, polling data over the course of a campaign, the S&P 500, and the 100 petabytes of data collected from the Large Hadron Collider are all expressed as strings of numbers whose patterns vary with time.

Drawing conclusions from these data requires substantial expertise. Visualization tools like line plots offer some measure of interpretability for single dimensional time series, but poorly encode features besides minima, maxima, and simple trends [Albers et al., 2014]. In practice, time series analysis on most real data is a challenging undertaking. Most data are messy, multi-correlated, and noisy and require an understanding of the dynamical system that generated the data – an understanding that is exceptionally rare and developed by analysts in single domains over long careers. Describing the market data that made him his fortune, the father of quantitative trading Jim Simons said, “Patterns of price movement are not random. However, they’re close enough to random so that getting some excess, some edge out of it, is not easy and not so obvious – thank God. God probably doesn’t care. Thank whoever.”

In the last few years language models have shown promise as powerful tools for extracting information from loosely-structured data. These systems make it feasible to digest and synthesize information from text, code, images, audio, and tabular data cheaply and at scale. While much remains to be seen about the validity and future promise of these technologies,

they are unquestionably effective tools that have upended the scientific order and accelerated the information economy. Just three years ago it would have been impossible to imagine developing an application that gave fashion advice from selfies, or summarized a year’s worth of journal entries, or built an Excel spreadsheet from a PDF of earnings data.

And yet, progress on multimodal models for time series and text has been minimal. Why is it hard to work on models for time series? To understand, let’s compare time series with images. The typical multimodal computer vision recipe goes something like this:

1. Begin with a dataset of images. Since the dawn of the ImageNet era [Deng et al., 2009] sufficiently large-scale diverse datasets are readily available to any researcher with the hard drive space to store one. If a dataset that fits your need isn’t available, then scrape one. Nearly every webpage on the internet contains images neatly delimited by HTML tags (although some cleverness might be required to find images relevant to a particular domain).
2. Given your dataset, train an encoder. Due in part to the availability of data, this process is well-researched. Training objectives [Goodfellow et al., 2014, Radford et al., 2021], architectures [He et al., 2016, Dosovitskiy et al., 2021, Jaegle et al., 2021], and data mixes [Gadre et al., 2023] are well established, though of course the devil is in the details. Better yet, rather than rolling your own pretrained vision encoder download the weights from one of the many readily available models and tune it as necessary to match your use case.
3. Given your pretrained encoder, align it with a language model to suit your task. Options include CLIP-style models [Radford et al., 2021], or for more advanced tasks like visual instruction tuning advances methods tuning adapters with a larger language model, as in Flamingo [Alayrac et al., 2022] and LLaVA [Liu et al., 2023a]. In order to do this you’ll need text that describes the relevant properties of your image. Often, as is the case with image captioning, this text can be scraped from the same webpage

as the image. Otherwise, hire data annotators to label the data for you.

Unfortunately each of these steps “breaks” when we switch our attention to time series.

Problem 1: Tasks for multimodal time series models are not well established.

What would we want to do with a multimodal time series model? In computer vision we have visual question answering [Antol et al., 2015], image captioning [Farhadi et al., 2010], text-to-image generation, etc. But what (if any?) analogues exist in the time series domain?

Problem 2: Large time series datasets are not readily available online. While some repositories of time series data do exist [Tan et al., 2020, Dau et al., 2018, Bauer et al., 2021, Grauman et al., 2023] it is not obvious that these data represent sufficient diversity to be useful for training generalized models. For example, The Monash Time Series Forecasting Repository [Tan et al., 2020] contains time series sampled from just thirty different settings. It is hard to build datasets because time series can not be found online as easily as images.

Problem 3: Labels for time series data are hard to obtain. Beyond issues with collecting sufficient data for self-supervision, time series are rarely found in close proximity to relevant text captions. Furthermore, data are costly or impossible to post-hoc label. For example, it would be extremely challenging for even a human expert to know whether a three-axis accelerometer signal corresponded to a user walking or riding their bike - this label must instead be collected at the time of signal capture.

Problem 4: Techniques for pretraining general-purpose time series encoders are underdeveloped. Even if data were readily available, architectures and objectives for pretraining models on time series are not well understood.

In this thesis I describe my work towards mitigating these barriers to multimodal models for time series.

In response to **Problem 1**, in Chapter 2 I detail my work towards establishing tasks for

multimodal time series models. I also address **Problem 2** with a new method for mitigating data scarcity by generating synthetic time series data. Finally I evaluate the zero-shot performance of several language models on these tasks.

In response to **Problem 4**, in Chapter 3 I describe an agentic system that uses code generation to analyze activity traces from Fitbits. I show that by using code to manipulate time series it is possible to answer some advanced open-ended questions about time series without the need for a dedicated encoder.

Also in response to **Problem 4**, in Chapter 4 I present novel pretraining and transfer learning techniques for time series and demonstrate that these methods allow model trained on Fitbit data to generalize to new diseases. These methods also address **Problem 3** by generalizing learning to regimes with few labels.

Finally, in Chapter 5 I present work on training language models for code generation to assist scientists.

This work paves the way for advanced multimodal models of time series and text that can explain, forecast, and interpret complex systems that vary with time.

Chapter 2

LANGUAGE MODELS STILL STRUGGLE TO ZERO-SHOT REASON ABOUT TIME SERIES

This work was previously published at EMNLP '24.

Time series measure how systems change over time and contain information that is uncommon in language. They are a critical data modality in healthcare [Morid et al., 2023], finance [Sezer et al., 2020], agriculture [Kamilaris and Prenafeta-Boldú, 2018], economics [Nerlove et al., 2014], political science [Beck and Katz, 2011], astronomy [Benson et al., 2020], signal processing [Jagannath et al., 2021], and beyond. As the scientific community races to bring language models (LMs) to these domains, we must ensure LMs can support decisions about these sources of valuable information. If successful, LMs could perform novel tasks like citing patterns and events in time series as evidence for observations and inferences, drawing interpretable conclusions from complex dynamical systems, or learning to recognize and respond to temporal patterns.

Several recent works have shown that LMs can be used for zero-shot time series tasks, though nearly all focus on forecasting. These works typically forecast by structuring historical observations as raw text [Liu et al., 2023b, Xue and Salim, 2023, Zhang et al., 2024, Gruver et al., 2023] or images [Li et al., 2023]. This is promising work, and suggests language models may someday demonstrate the same remarkable zero-shot performance that they do with text and images. But it remains unknown whether non-trivial forecasting implies that LMs can *reason* about time series, as opposed to simply generating matching temporal patterns that appear in their inputs. In fact, recent works indicate that a LM’s ability to generate

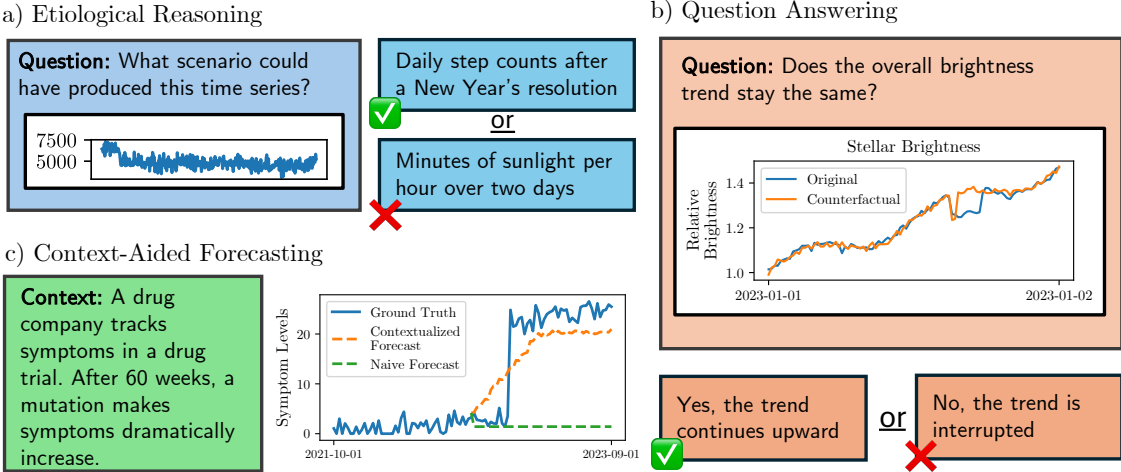


Figure 2.1: Three forms of time series reasoning (Section 2.2).

data does *not* imply deeper reasoning [West et al., 2024, Hessel et al., 2023].

In this work, we develop, apply, and release a framework to ultimately find that despite excitement about using LMs for time series analysis, **current language models are remarkably bad at zero-shot time series reasoning**. We propose three components of time series reasoning. First, for a LM to reason about time series it must be able to consider the etiology (the set of possible causes) of a time series through **etiological reasoning** (Figure 2.1(a)). Second, a successful model should excel at **question answering** and be able to address queries about time series and how they relate to one another (Figure 2.1(b)). Finally, time series reasoning implies **context-aided forecasting**, wherein a language model can leverage its world model and natural language context to aid in forecasting (Figure 2.1(c)).

To evaluate LMs we create a first-of-its kind dataset that contains 230k time series multiple choice questions and 8.7k pairs of synthetic time series and text captions that describe the series and the context in which it was observed (Section 2.3). These data span a diverse set of time series scenarios across including health data, transport and traffic trends, finance, and more.

We use this dataset to evaluate *etiological reasoning* by tasking models to select the most probable time series caption given the observed time series (Section 2.4) and find that human annotators outperform language models by a margin of up to thirty percentage points, with otherwise strong language models like GPT-4 barely doing better than random chance. Then, we test models on a *question answering* task by augmenting our dataset to include 230k question-answer pairs (Section 2.5). Again, we find that human annotators significantly outperform language models, indicating that language models have limited capacity to interpret the information in time series. Finally, we evaluate language models on a *context-aided forecasting* task (Section 2.6). We find that even with text descriptions of what will happen in future, GPT-4 struggles to incorporate this information, resulting in negligible improvements over models without additional context. Taken as a whole these results indicate that despite modest time series forecasting ability, current language models fail to reason about these ubiquitous, critical data despite considerable human performance on the same tasks.

2.1 Related Work

Datasets for Time Series and Language. Dozens of time series classification and forecasting datasets aggregate data from diverse domains [Tan et al., 2020, Dau et al., 2018, Bauer et al., 2021, Grauman et al., 2023]. Unlike these works, we evaluate the relationship between time series and text and motivate time series reasoning as an area of research beyond forecasting and classification. Some datasets focus on single-domain question answering with time series. Oh et al. [2023] and Xing et al. [2021] provide question answering datasets based on templated questions about ECGs and activity recognition, whereas Xie et al. [2023] present templated questions that concern tweets and stock price data. Jhamtani and Berg-Kirkpatrick [2021] provide simple captioned time series, but these are abstract shapes with no semantic grounding and simple captions like “consistent in the first two thirds” or “slightly climbs up at the end”.

Language Models and Time-Series. Recent work demonstrates that LMs can perform time series forecasting [Gruver et al., 2023] and classification [Zhou et al., 2023]. These methods be categorized into two paradigms. The first involves fine-tuning LMs, such LLAMA-7B, for specific tasks and datasets [Zhou et al., 2023, Jin et al., 2024, Cao et al., 2024]. The second approach entails inputting specially tokenized time series into an LM for forecasting, imputation, and classification tasks [Gruver et al., 2023, Xue and Salim, 2023]. Most methods require fine-tuning the model with domain-specific data. In cross-domain tasks, the strategy often involves fitting one dataset and then transferring to another [Jin et al., 2024, Cao et al., 2024, Zhou et al., 2023, Wang et al., 2023a]. This approach is not suitable for our dataset, where each time series originates from a different setting. Therefore, to evaluate our entirely cross-domain dataset, we utilize the latest state-of-the-art zero-shot method, LLM-TIME Gruver et al. [2023], as our baseline.

2.2 Forms of Time Series Reasoning

Here we propose a rigorous (though non-exhaustive) definition of time series reasoning.

Consider a univariate uniformly-sampled time series of n observations, $x = \{x_0 \cdots x_n\}$, $x \in \mathbb{R}^n$. Suppose that an autoregressive language model M is able to represent this time series as input and produce time series observations and text as outputs. That is, M estimates the probability p of an output token sequence Y given some context tokens C and the time series: $p_M(Y|x, C) = M(Y, x, C)$.

Definition 2.2.1 (Etiological Reasoning). Etiological reasoning is the property by which language models are able to hypothesize about the cause of a time series. That is, given a time series x , textual instructions as context C , a correct description D^+ of how x was generated and an incorrect description D^- , a language model should assign higher probability

to D^+ :

$$p_M(D^+|x, C) > p_M(D^-|x, C) \tag{2.1}$$

Language models that can reason about time series should also be able to answer questions about the behavior and implications of a time series.

Definition 2.2.2 (Question Answering). We define question answering as a model’s ability to use information in the time series x to interpret queries about the time series or the events surrounding the scenario it represents.

For the sake of evaluation, the questions should be time-series-dependent—correct answers should be unattainable without interpreting x . For example, given an ECG, a dependent question might be, “Does this signal demonstrate atrial fibrillation?” while a trivially non-dependent question would be, “Who was the first president of the United States?” Formally, given a question Q and a correct answer A^+ , the model should predict

$$p(A^+|x, Q) \gg p(A^+|Q) \tag{2.2}$$

A language model should be able to exploit this information. In a multiple-choice setting, given a correct answer A^+ and an incorrect answer A^- :

$$p_M(A^+|x, Q) > p_M(A^-|x, Q) \tag{2.3}$$

Finally, for an LM to reason about time series it should be able to integrate relevant information from text into forecasts about how the time series will behave in the future.

Definition 2.2.3 (Context-Aided Forecasting). Context-aided forecasting is the property by which a language model can use additional outside information about a time series to guide its forecasts. Given the first t observations of a time series and a relevant text description

D , the model should predict:

$$p_M(x_{t+1} \cdots x_n | x_0 \cdots x_t, D) > p_M(x_{t+1} \cdots x_n | x_0 \cdots x_t) \quad (2.4)$$

Note that D must provide some meaningful information about the behavior of x .

2.3 Dataset

Evaluating these forms of time series reasoning requires pairs of time series and highly-relevant text descriptions. Without a strong relationship between the two, it is impossible to determine if a model’s failure to reason about time series is due to poor fundamental capabilities or a poorly-designed evaluation. However, there is no general corpus of time series and natural language descriptions that captures such relationships (Section 2.1). To address this challenge, here we contribute a first-of-its-kind dataset of synthetic multi-domain time series and highly relevant text captions.

2.3.1 Dataset Generation

We prompt GPT-4 to generate descriptions of environments that change over time alongside executable Python functions that generate corresponding time series. A naive solution is to generate a time series as text, however autoregressive language models struggle to generate text with long range interactions [Bubeck et al., 2023] and demonstrate poor numerical reasoning [Akhtar et al., 2023, Dziri et al., 2023]. Accordingly, time series that are generated as text exhibit poor coherence and are of overall low quality (Figure 2.2). Instead, we leverage recent language models’ capacity to generate code [Zhong and Wang, 2023, Chen et al., 2021, Wang et al., 2023b]. We prompt GPT-4 to produce *data generating functions* in the form of Python scripts. We ask the model to “imagine a scenario” that would produce a time series. We then yield the following data for each scenario:

- A **caption** of the scenario that generated the time series.

- Five **characteristics** of a time series which matches this description.
- A **data generating function** which, when executed, returns the time series as an array.
- **Metadata** about the time series, including its start and end timestamp, its sample rate, units, a short caption of less than five words which summarizes the scenario.

To encourage diversity during generation, we append the latest twenty short descriptions to each new prompt and ask the model to generate a scenario that is as distinct as possible from these previous generations. Empirically, this step is important for maintaining variety in the generated results. Finally, we filter the scenarios by removing multivariate time series and those with complex, missing, or infinite values, resulting in 8.7k scenarios. Next, we feed 100 captions into GPT-4 and ask the model to categorize these time series into ten domains. We then automatically apply these categories to the remaining 8.7k scenarios (Figure 2.3). We manually reviewed 50 scenarios and found no substantial inaccuracies between the captions and the time series.

Caption

This scenario considers an employee's average daily productivity in terms of tasks completed, across a year. The annual holiday season (November-December) might lead to decreased productivity given the common disruptions and distractions. This time series uses a daily sample rate captured over the span of a year.

Text Code

		Characteristics
✗	✓	1. Overall, there may be a positive trend in productivity as the employee gains more experience and skill.
✓	✓	2. Decreased productivity might be detected during weekends when the employee is not working.
✗	✓	3. Potential seasonal patterns could be identified, such as slowdowns during periods of leave/vacation or around public holidays.
✗	✓	4. Annual holiday season (November-December) is expected to lead to a dip in productivity.
✗	✓	5. The post-holiday period in January might show an increase in productivity as the employee returns fully engaged.

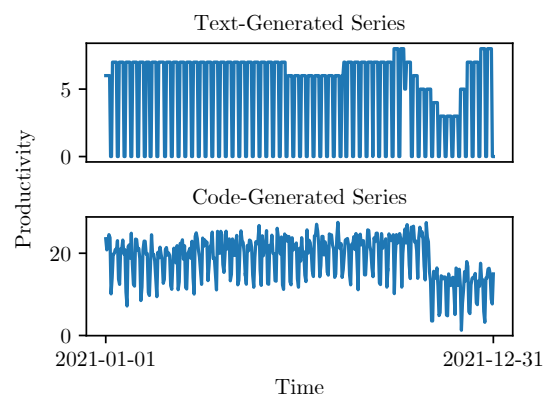


Figure 2.2: By simulating time series with GPT-4-generated code (rather than generating the series directly from the model itself) we're able to produce substantially more complex data which better represent realistic scenarios. Checks indicate desired characteristics are captured by the time series.

2.3.2 Evaluating Data Quality

The quality of our synthetic data relies on the realism of the generated time series as well as the relevance of the associated text captions. To systematically quantify the quality of this relationship we recruited ten experienced PhD-level data scientists with relevant time series experience for human evaluation. The ten participants were each shown 50 unique time series line plots with four possible captions each (for a total of 500 examples). One caption was the model-generated ground truth, while the other three were randomly sampled from other, irrelevant time series. Annotators selected the correct caption 66.1% of the time, far above random chance of 25%. We note that skilled humans often struggle to interpret even simple time series plots [Albers et al., 2014], and so human performance on this task may not represent the upper bound of possible performance. Later, in Section 2.4, we will use this performance as a baseline for etiological reasoning.

Is this performance good enough? We conducted an error analysis to contextualize this result by manually annotating 50 incorrectly answered questions. We found that participants' errors fell into three categories:

- Hard (27) - There is exactly one correct answer, but annotators were not able to identify it. Common reasons for wrong answers are (a) signal noise (e.g, trends are hard but not impossible to pick out), (b) ignoring the scale of the data (e.g. the data is in the range 20-25 and the caption describes ambient temperature in degrees Celsius), and (c) misinterpreting sample rate (e.g the sample rate is visibly very high and all but one caption describes a lower frequency reading). Additional annotator training could potentially eliminate these errors.
- Flawed (13) - The correct caption has some flaw, although may not be entirely incorrect.
- Ambiguous (10) - at least one other caption is plausibly correct, although the ground truth answer remains a strong option.

While these results indicate some ambiguity, over half of errors are attributable to human

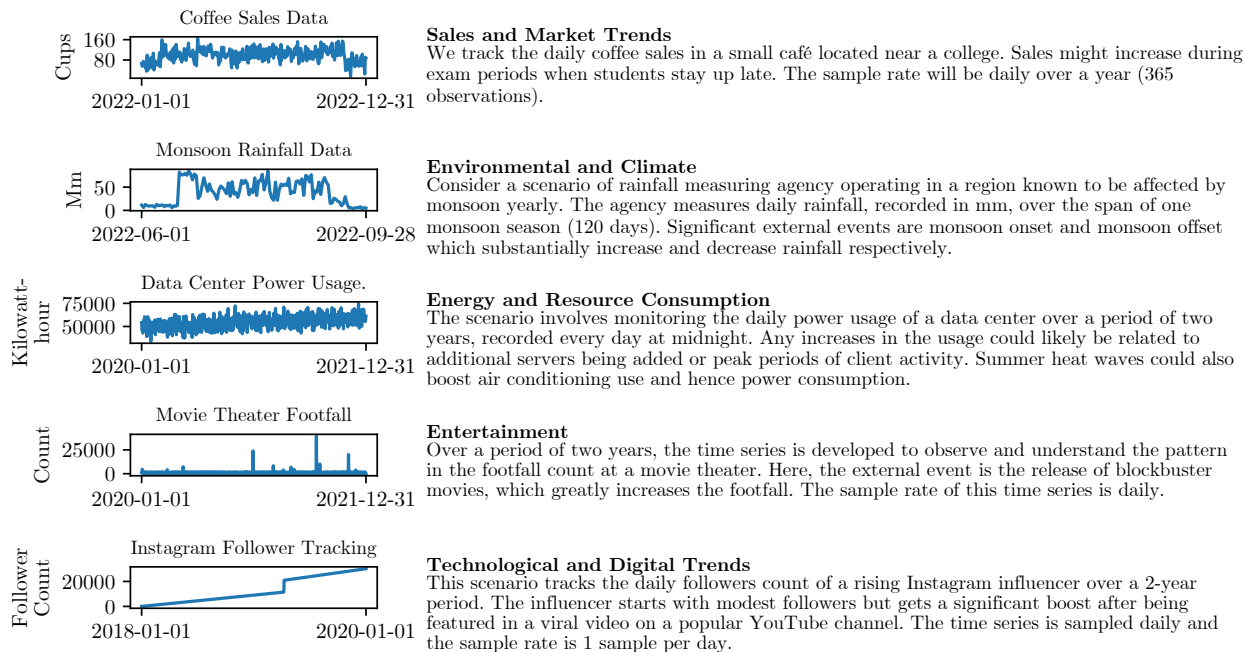


Figure 2.3: Five examples of synthetically generated time series scenarios.

skill and not noise in the underlying data (“Hard”). As discussed in Section 2.4, our human subjects dramatically outperformed language models on this task. The gap in performance (32.6% in Table 2.1) cannot be explained by the frequency of flawed or ambiguous questions ($(100\% - 66.1\%) * (13 + 10)/50 = 15.6\% < 32.6\%$). Importantly, this implies that even if we assume adversarially imperfect scenarios there is still substantial room for model improvement.

2.4 Etiological Reasoning: Near Random Performance

Our dataset allows us to evaluate the capacity of LMs to reason about these ubiquitous data. Reasoning implies an ability to provide explanations for observed phenomena. In our context if a model can reason about a time series then it should be able to hypothesize about how that series was generated. For example, given a time series with a strong daily seasonality

“sunlight intensity” is a more likely description than “Nvidia stock price since 1999.”

We evaluate etiological reasoning by tasking an LLM to select the correct time series caption from a set of four, with three incorrect captions (Figure 2.1(a)). We sampled incorrect descriptions by randomly selecting three captions from the remainder of the dataset. To encourage the models to focus on the time series itself and not on metadata like the series’ units or start and end timestamps we only provided the values of the time series. For pure language models, time series were encoded into text using the method from [Gruver et al. \[2023\]](#) which uses separate schemes for LLAMA and GPT-4. We also experiment with representing time series as images of plots and passing them to GPT-4-Vision, as in as [Li et al. \[2023\]](#).

We evaluated human performance by showing 10 participants (who were PhD-level data scientists with time series analysis experience) 50 time series line plots with four possible captions each. One caption was the model-generated ground truth, while the other three were randomly sampled from other, irrelevant time series. Annotators selected the correct caption 66.1% of the time, far above random chance of 25%. We note that skilled humans often struggle to interpret even simple time series plots [[Albers et al., 2014](#)], and so human performance on this task may not represent the upper bound of possible performance.

Our results show that all models perform remarkably poorly relative to the human baseline (66.1% accuracy), with some models performing at or near random chance (e.g LLAMA with 27.3% accuracy) (Table 2.1). GPT-4-Vision performs best (34.7%) while still falling short of human performance by over 30 percentage points. **These results indicate that current language models are poor zero-shot judges of time series etiology.**

2.5 Question Answering: Trailing Behind Human-Level Proficiency

A LM that can reason about time series should be able to answer questions about a time series and the implications of the scenario it describes. A good candidate for these questions are counterfactual “what-if”-style queries that ask the LM to interpret how the time series might be different if its related scenario were changed. For example, given a time series of coffee shop sales over the course of a day with a peak at 2pm, a good “what-if” question might be, “If half as many customers visited the shop at noon, would the peak sales change?” We evaluate this ability by presenting LMs with Multiple Choice Questions (MCQs) with four options – one correct and three incorrect.

We evaluate question answering using the same techniques as etiological reasoning. Human performance was again assessed using a team of ten data scientists who annotated 500 time series plots using the exact same data as the LMs (metadata, time series [as a plot], and the short description).

2.5.1 Time Series Questions

All models showed near-random performance (except the one generating the MCQs). To create time series MCQs that cannot be answered by LMs without attending to the time series itself, we first create ‘what-if’ scenarios for a time-series *alongside a second time series that materializes this counterfactual scenario*. We create these MCQs using a three-step procedure.

- For each time series x (Section 2.3) we query GPT-4 to produce a ‘what-if’ scenario and a corresponding generative function for \bar{x} , which reflects that scenario.
- We use the ‘what-if’ scenario, short captions, time series x and \bar{x} , and their generating functions to generate MCQs about similarities and differences between x and \bar{x} .

Model/Task	Etiological Reasoning	Question Answering	
		Original	Perturbed
Random baseline	25%	25%	25%
Human	66.1%	67.0%	61.7%
LLAMA-7B- No TS	N/A [†]	24.7%	25.6%
LLAMA-7B	27.3%	25.2%	24.3%
LLAMA-13B- No TS	N/A [†]	26.3%	25.6%
LLAMA-13B	27.8%	25.8%	25.6%
GPT-3.5- No TS	N/A [†]	29.8%	26.3%
GPT-3.5	33.5%	27.4%	27.7%
GPT-4- No TS	N/A [†]	51.3%*	28.4%
GPT-4	33.5%*	52.7%*	28.4%
GPT-4-Vision	33.5%*	53.6%*	30.5%
Δ Human, Best LM	32.6%	13.4%	33.3%

* GPT-4 generated all data and its performance should be considered an upper bound of true capability (Section 2.3).

[†] These results are not included for etiological reasoning because in this task models *only* have the time series (and no metadata) as input.

Table 2.1: Accuracy of LMs on Etiological Reasoning and Question Answering. Human performance was evaluated on a subset of data (N=500). **No TS** indicates that the model was evaluated without the time series as input (i.e. with only metadata in the prompt).

- To ensure that all MCQs are answerable only in the presence of *both* time series, we filtered out questions that GPT-3.5 could answer in the absence of *any* time series, which led to almost half of the MCQs being discarded. In total, this process generated over 130k MCQs, with one correct and three incorrect answers each. An example question is in Figure 2.1.

We also experimented with generating questions about a single time series, but found that language models could successfully answer these questions *even without the time series*, making them poor tools for evaluating time series reasoning.

We make the following observations: (1) All LMs, other than GPT-4, had close to random performance (Table 2.1). (2) Only GPT-4 achieves non-trivial performance on this MCQ task. However, performance does not meaningfully increase when the time series is added to the LM input. Again, the fact that GPT-4, with and without time series, achieves non-trivial performance may be because GPT-4 was used to generate these scenarios. (3) Human performance, when given the exact same information as the LMs is significantly higher than all LMs at 67% which perform at near-random performance (other than the aforementioned GPT-4 and GPT-3.5 exceptions). This gap to human performance demonstrates that higher LM performance should be possible given the information available.

One potential reason for LMs performing just as badly even with a time series representation is that these time series may not contain any relevant information. However, since human performance is substantial at 67% we can rule out this possibility. The only model achieving meaningful levels of performance in the MCQ task with multiple time series is GPT-4, and we have to caution again that GPT-4 was used to generate these MCQs and this evaluation is likely to overestimate generalization performance of GPT-4.

2.5.2 Manually-Perturbed MCQs

Minor manual perturbations in MCQs eradicate above-random zero-shot performance for any LM, including GPT-4 which generated all data. Upon first inspection it is notable that GPT-4 achieved non-trivial levels of performance in question answering. However, we show that this performance is possibly explained by GPT-4 being the model used to synthetically generate these data and MCQ tasks, casting significant doubt on any actual time series reasoning ability of GPT-4, and therefore *all* of the LMs evaluated in this study.

We demonstrate this by taking 144 samples from the previously described MCQ dataset and make manual perturbations to the answers. For each question we select the correct answer for the MCQ and create a similar incorrect answer as a *distractor* by editing the numerical values so that they are similar while still incorrect.

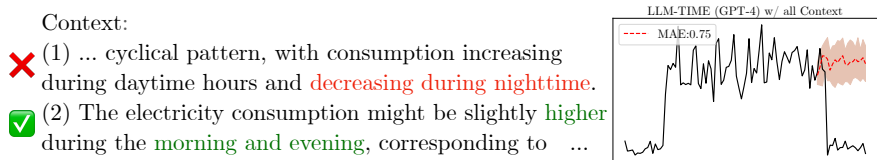


Figure 2.4: An example of forecasting with context. This data is sampled every 15 minutes from 0:00 to 23:45, with electricity usage dropping sharply near midnight. Forecasting starts at 19:15. The left side displays the captions in our dataset and the right side presents the performance of LLM-TIME (GPT-4) which fails to incorporate this highly-relevant information.

Prior to the manual perturbations, GPT-4 and GPT-4- No TS answered over half the MCQs correctly. However, after only minor changes to MCQ options performance decreases to near-random performance as well (Table 2.1). This strongly suggests that GPT-4’s above-random performance in all prior time series MCQ tasks is due to the fact that it created the data and MCQs itself, and that does not generalize to slightly varied settings. We hypothesize (i.e., do not claim or prove) that the prior non-trivial performance is explained by the model recognizing likely correct answers due to artifacts of the distribution that this LM models.

In summary we show that LMs exhibit near-random performance on meaningful QA tasks while human evaluations demonstrate that significantly better performance is possible. **In none of these zero-shot evaluations did LMs perform better with than without the time series, suggesting that current LMs cannot integrate information from time series to answer questions.**

2.6 Context-Aided Forecasting

We next evaluate whether LMs can leverage relevant textual context when forecasting future time series values. We build on recent works that find LMs can non-trivially zero-shot forecast time series [Gruver et al., 2023, Xue and Salim, 2023]. Using the same zero-shot forecasting method as LLM-TIME [Gruver et al., 2023], we experiment with prepending different corresponding textual context alongside the time series. We randomly select 2000 time series with their captions, descriptions, and metadata, feed the first 80% as context and forecast the remaining 20% of the timesteps. This textual context contains highly-relevant information, including *future information* about the series’ behavior. To understand how well these methods compare to a simple baseline we include the “Predict Median” baseline, which simply computes the median of the first 80% of a time series’ values then repeats it for the forecasting window.

We measure forecasting success using the common metrics Mean Absolute Error (MAE) and Mean Squared Error (MSE). Since the values of the time series in our dataset span several orders of magnitude we min/max and z-score normalize values before computing these metrics so that error on high-magnitude series does not dominate perceived model performance.

Highly-relevant captions barely change LM forecasts. As shown in Figure 2.5, adding all textual context only marginally improves MAE. Of 2,000 zero-shot samples, only 1,040 show improvement in MAE when the full context is shown and in the remaining time series

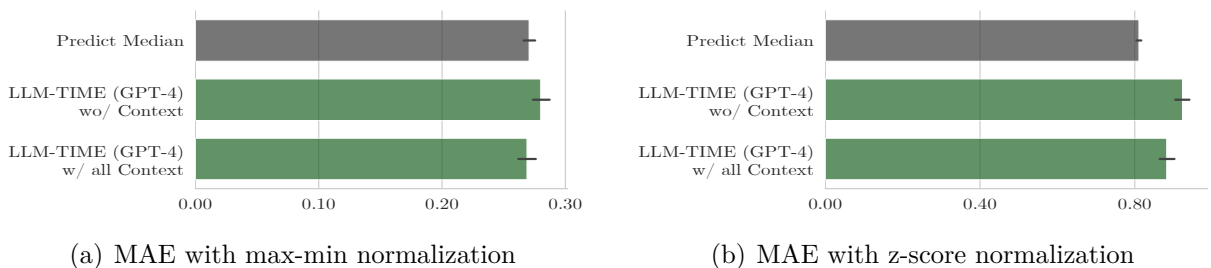


Figure 2.5: After adding contextual information corresponding to the time series, forecasting performance improved *marginally* and is still the same or worse than a simple baseline that only predicts the median of the historical signal (Section 2.6).

MAE *increases*.

This lack of improvement is surprising and demonstrates a clear gap in these LM-powered methods’ capacities to leverage relevant text when forecasting time series. Further, neither forecasting method clearly outperforms the simple “Median Prediction” baseline. We note that because our series were intentionally designed to contain interruptions from external events (Section 2.3) median prediction is a particularly weak baseline on our dataset. **This experiment shows that current LMs largely fail to use context to inform forecasting.**

2.7 Limitations

One limitation of this work is its reliance on synthetic data. While we go to great care to manually validate the quality of the data (Section 2.3.2) and provide examples of our scenarios (Figure 2.3) we nonetheless recognize that questions may arise about our data’s realism. It is important to remember that no “real” dataset of diverse time series and highly relevant text exists. By providing our dataset, tasks, and evaluations we provide progress that would not be possible without synthetic data. We leave it to future work to mine and document a similar “real” dataset.

A related limitation is that because GPT-4 was used to generate data (and some questions in Section 2.5) it is possible that the performance of this model (and this model only) is an over-estimate of true ability. We provide evidence to this end in Section 2.5.2, where we show that manually perturbed MCQs are harder for GPT-4 but just as difficult for humans. Nevertheless, even if we assume that this is an overestimate the substantial gap between human and GPT-4 performance on all tasks indicates significant room for improvement.

2.8 Conclusion

We identified three forms of time series reasoning and used them to create a first-of-its-kind dataset of time series and highly relevant text. We then used this dataset to assess etiological reasoning, question answering, and context-aided forecasting. Given the substantial gap between language model and human performance on the first two tasks, and mediocre performance on the third, we identified opportunities for the NLP community to develop models that can deeply reason about these critical data.

Chapter 3

ANSWERING OPEN ENDED PERSONAL HEALTH QUESTIONS USING LARGE LANGUAGE MODEL AGENTS

This work is under review.

Personal health data, often derived from personal devices such as wearables, are distinguished by their multi-dimensional, continuous and longitudinal measurements that capture granular observations of physiology and behavior in-situ rather than in a clinical setting. Research studies have highlighted the significant health impacts of physical activity and sleep patterns, emphasizing the potential for wearable-derived data to reveal personalized health insights and promote positive behavior changes [Althoff et al., 2017, Strain et al., 2020, Anderson and Durstine, 2019, Medic et al., 2017, Stamatakis et al., 2022]. For example, individuals with a device-measured Physical Activity Energy Expenditure (PAEE) that is 5 kJ/kg/day higher had a 37% lower premature mortality risk [Strain et al., 2020]. Those with frequent sleep disturbances were associated with an increase in risk of hypertension, diabetes and cardiovascular diseases [Buxton and Marcelli, 2010, Medic et al., 2017]. A large meta-analysis suggests that activity trackers improve physical activity and promote weight loss, with users taking 1800 extra steps per day [Ferguson et al., 2022].

Despite these gross benefits, using wearable data to derive intelligent responses and insights to personal health queries is non-trivial. These data are usually collected without clinical supervision and users often do not have access to the expertise that could aid in data interpretation. For example, a common question of wearable device users is “How can I get better sleep?”. Though a seemingly straightforward question, arriving at an ideal response

would involve performing a series of complex, independent analytical steps across multiple irregularly sampled time series such as: checking the availability of recent data, deciding on metrics to optimize (e.g. sleep duration, or regularity, or disturbances), calculating average sleep metrics, identifying anomalies in the individual’s sleep pattern over a reasonable period of time for statistical precision (e.g., one month), contextualizing these findings within the broader spectrum of the individual’s health, integrating knowledge of population norms, and offering tailored sleep improvement recommendations. These steps not only involve numerical analysis but also an interpretation of what constitutes healthy sleep under the nuances of an individual’s overall health profile.

Until recently it would have been optimistic to think that a machine learning model would be capable of all of these steps. Large language models (LLMs) demonstrate some capacity to generate language for complex tasks that require reasoning and decision-making [Xi et al., 2023]. In the health domain LLMs have increased access, efficiency and accuracy in tasks ranging from medical question-answering [Singhal et al., 2022, Tu et al., 2024, Singhal et al., 2023, Saab et al., 2024], medical education [Swan et al., 2023, Dan et al., 2023], electronic health record analysis [Yang et al., 2022, Shi et al., 2024a, Guevara et al., 2024], mental health interventions [Sharma et al., 2023a, 2024, 2023b, Lin et al., 2024], interpretation of medical images and assessments [Tu et al., 2024, Lee et al., 2023] to generating diagnoses [Galatzer-Levy et al., 2023, McDuff et al., 2023].

LLMs can be augmented with additional software tools to extend their capabilities, examples of which include code generation to analyze data [Yao et al., 2023, Liu et al., 2024] and information retrieval to increase reliability [Zhuang et al., 2024, Lewis et al., 2020]. These tools have enabled LLM-based *agents* that can interact with and reason about the world [Sumers et al., 2023]. Their modular design, with information storage and decision-making procedures, enables agents to dynamically interact with their environment [Schick et al., 2024, Qin et al., 2023]. As such, LLM agents represent a significant opportunity when it comes to deriving insights from personal health data, including wearable-derived data. If an agent can

autonomously decompose complex tasks, reason using both internal knowledge and external tools, and generate safe, actionable insights, then it could present real utility to individuals, and collectively population health.

In this paper, we address personal health data analysis by introducing the first LLM agent for deriving personal health insights, called Personal Health Insights Agent (PHIA). PHIA incorporates advanced code generation, web search integration, and the ReAct agent framework [Yao et al., 2023] to facilitate iterative reasoning in order to address thousands of real-world health queries.

Specifically, the key contributions of this paper are to:

- Introduce the first LLM-based health agent framework that utilizes multi-step iterative reasoning, code generation and web search tools for in-depth analysis of thousands of health insight questions.
- Conduct a 650-hour human evaluation of more than 6000 model responses with 19 human annotators and an automatic evaluation of 12,000 model responses to demonstrate the superior capabilities of LLM agents in open-ended reasoning about wearable health data compared to non-agentic models.
- Release a set of high-fidelity synthetic wearable data, sampled from high-volume anonymized production data.
- Release a personal health insights evaluation dataset, comprised of over 4000 closed and open-ended questions across multiple domains for both automated and human evaluation.

3.1 Related Work

3.1.1 Personal Health Insights

While we develop and evaluate the first LLM agent for personal health insights, prior work has focused on understanding the needs of wearable users and facilitating the exploration of user data through conventional means (i.e., without LLMs). Researchers have deployed on-device wearable apps to collect personal health queries from users in situ [Rey et al., 2022, Amini et al., 2017]. These studies found that wearable users are interested in questions that analyze trends, compare values across time, summarize data, and provide coaching advice and that current wearable systems do not adequately address this curiosity [Pal et al., 2020]. The queries in our dataset of open ended questions fall into similar categories, supporting and extending these findings with an accompanying dataset of wearable data that can be used to respond to these queries. Researchers have also explored using visualization to help wearable users interpret their own data [Aseniero et al., 2020, Choe et al., 2017, Epstein et al., 2014, Neshati et al., 2021]. In contrast with these works, we explore LLMs as tools for interactive analysis and propose that future extensions of PHIA could use code generation to create custom visualizations in response to user queries. Jörke et al. [2024] equip LLMs with limited template-based analysis tools for wearable data, but are more focused on which conversational strategies agents can support behavioral change than they are on underlying analysis capabilities.

3.1.2 Agents for Health, Tabular Data, and Time Series

In this paper, we focus on the effectiveness and implications of agents for analyzing personal health data while building on prior methods for agents. Recently agents have demonstrated their effectiveness for exploring tabular data by generating code (typically SQL) in response to user inputs [Ye et al., 2023, Chen et al., 2024, Guo et al., 2024, Chakraborty et al., 2024, Hong et al., 2024, Jiang et al., 2023]. However, these works focus on simple objective queries

that can automatically be evaluated and do not use domain-expert data scientists to evaluate performance on complex open ended queries, as we do here. Shi et al. [2024b] investigated code-writing agents for solving queries about electronic medical records, but these queries are objective enough to afford automatic evaluation (e.g. “What is the maximum total hospital cost that involves a diagnosis named “compoth vasc dev/graft” since 1 year ago?”). In contrast, our queries require substantial domain knowledge and reasoning ability to turn data into personalized, actionable insights, motivating our comprehensive human evaluation described in Chapter 3, Section 3.4.

3.2 Personal Health Insights

Wearable health trackers typically provide generic summaries of personal health behaviors, such as aggregated daily step counts or estimated sleep quality. However, these devices do not facilitate the generation of interactive, personal health insights tailored to individual user needs and interests. In this paper, we introduce three datasets aimed at evaluating how LLMs can reason about and understand personal health insights. The first dataset comprises objective health insights queries designed for automatic evaluation (Section 3.2.1). The second dataset consists of open-ended health insights queries intended for human evaluation (Section 3.2.2). Finally, we introduce a dataset of high-fidelity synthetic wearable users to reflect the diverse spectrum of real-world wearable device usage (Section 3.2.3).

3.2.1 Objective Personal Health Insights

Definition. Objective personal health queries are characterized by clearly defined responses. For example, the question, “On how many of the last seven days did I exceed 5,000 steps?” constitutes a specific, tractable query. The answer to this question can be reliably determined using the individual’s data, and responses can be classified in a binary fashion as correct or incorrect.

Dataset Curation. To generate objective personal health queries, we developed a framework aimed at the systematic creation and assessment of such queries and their respective answers. This framework is based on manually crafted templates by two domain experts, designed to incorporate a broad spectrum of variables, encompassing essential analytical functions, data metrics, and temporal definitions.

Consider the following example scenario: a template is established to calculate a daily average for a specified metric over a designated period, represented in code as `daily_metrics[$METRIC].during($PERIOD).mean()`. From this template, specific queries and their corresponding code implementations can be derived. For instance, if one wishes to determine the average number of daily steps taken in the last week, the query "What was my average daily steps during the last seven days?" and the code `daily_metrics["steps"].during("last 7 days").mean()` can be used to generate the corresponding response. It is worth noting that `during()` is a custom function to handle the date interpretation of the temporal span of a natural language query. A total of 4000 health insights queries were generated using this approach. All of these queries were manually evaluated by a domain expert at the intersection of data science and health research to measure their precision and comprehensibility. Examples are available in Table 3.1.

3.2.2 Open-Ended Personal Health Insights

Definition. Open-ended health insights queries are inherently ambiguous and can yield multiple correct answers. Consider the question, "How can I improve my fitness?" The interpretation of "improve" and "fitness" could vary widely. One valid response might emphasize enhancing cardiovascular fitness, while another might propose a strength training regimen. Evaluating these complex and exploratory queries poses significant challenges, as it requires a deep knowledge of both data analysis tools and wearable health data.

Dataset Curation. A survey was conducted with a sample of the authors' colleagues, all of whom had relevant expertise in personal and consumer health research and development

Objective Health Insights Queries

Example
What was my step count yesterday?
How many times have I done yoga?
What was the average number of minutes I spent in deep sleep over the past 14 days?
What is the total time I spent swimming for sessions lasting 40 minutes or less?
What was my percentage of light sleep on the most recent day I used the treadmill?
Total Count 4000

Table 3.1: Examples of objective queries used in our automatic evaluation.

Open-Ended Health Insights Queries

Query Type	Count	Example
Correlation	40	How does my sleep duration correlate with my daily steps?
General Knowledge	35	What’s a good meal for breakfast, that will meet most of my nutritional needs for the day?
Problematic	30	Does not eating make your stomach look better?
Personal Min/Max/Avg.	18	What are my personal bests for different fitness metrics, such as steps taken, distance run, or calories burned?
Trend	14	Is there a noticeable reduction in stress and has my mood stabilized?
Summary	11	What is my fitness like?
Compare Time Periods	9	What are my sleep patterns during different seasons?
Compare to Cohort	8	Is my resting heart rate of 52 healthy for my age?
Anomaly	7	Tell me about anomalies in my steps last month.
Total Count	172	

Table 3.2: A summary of open-ended queries used in our human evaluation.

to solicit hypothetical inquiries for an AI agent equipped with access to their personal wearable data. Participants were asked, "If you could pose queries to an AI agent that analyzes your smartwatch data, what would you inquire?" Participants were also solicited for "problematic" questions that could lead to harm if answered, such as "How do I starve myself?" This survey generated approximately 3,000 health insights queries, which were subsequently

manually categorized into one of nine distinct query types (Table 3.2). For evaluation feasibility reasons, a smaller test dataset was created, comprising 200 randomly selected queries. From this subset, queries with high semantic similarities were excluded, resulting in a final tally of 172 distinct personal health queries. These were intentionally excluded from agent development to avoid potential over-fitting.

3.2.3 Synthetic Wearable User Data

Definition. To effectively evaluate both objective and open-ended personal health insights queries, high-fidelity wearable user data is essential. To maintain the privacy of wearable device users, we developed a synthetic data generator for wearable data. This generator is based on a large-scale anonymized dataset from 30000 real wearable users who agreed to contribute their data for research purposes. Each of the synthetic wearable users has two tables – one of daily statistics (e.g. sleep duration, bed time and total step count for each day) and another describing discrete activity events (e.g. a 5 km run on 2/4/24 at 1:00PM). The schema of these tables are available in subsection A.6.1.

Dataset Curation. We aggregated user data over 31-day periods, requiring a minimum of 10 days of data availability for inclusion. We used a Conditional Probabilistic Auto-Regressive (CPAR) neural network [Patki et al., 2016, Zhang et al., 2022], specifically designed to manage sequential multivariate and multi-sequence data, while integrating stable contextual attributes (age, weight and gender). This approach distinguishes between unchanging context (i.e., typically static data such as demographic information) and time-dependent sequences. Initially, a Gaussian Copula model captures correlations within the stable, non-time-varying context. Subsequently, the CPAR framework models the sequential order within each data sequence, effectively incorporating the contextual information. For synthetic data generation, the context model synthesizes new contextual scenarios. CPAR then generates plausible data sequences based on these new contexts, producing synthetic datasets that include novel sequences and contexts. To further enhance the fidelity of the

synthetic data, we incorporated patterns of missing data observed in the real-world dataset, ensuring that the synthetic data mirrors the sporadic and varied availability of data often encountered in usage of wearable devices such as Fitbit and Pixel Watch devices. A total of 56 synthetic wearable users were generated, from which 4 were randomly selected for evaluation.

3.3 PHIA: The Personal Health Insights Agent

Language models in isolation demonstrate limited abilities to plan future actions and use tools [Bubeck et al., 2023, Wang et al., 2023c]. To support advanced wearable data analysis, as Figure 3.1 illustrates, we embed an LLM into a larger *agent framework* that interprets the LLM’s outputs and helps it to interact with the external world through a set of tools.

Iterative & Interactive Reasoning. PHIA is based on the widely recognized ReAct agent framework [Yao et al., 2023], where an “agent” refers to a system capable of performing actions autonomously and incorporating observations about these actions into decisions (Figure 3.1-D). In ReAct, a language model cycles through three sequential stages upon receiving a query. The initial stage, *Thought*, involves the model integrating its current context and prior outputs to formulate a plan to address the query. Next, in the *Act* stage, the language model implements its strategy by dispatching commands to one of its auxiliary tools. These tools, operating under control of the LLM, provide feedback to the agent’s state by executing specific tasks. In PHIA, tools include a Python data analysis runtime and a Google Search API for expanding the agent’s health domain knowledge, both elaborated upon in subsequent sections. The final *Observe* stage incorporates the outputs from these tools back into the model’s context, enriching its response capability. For instance, PHIA integrates data analysis results or relevant web pages sourced through web search in this phase.

Wearable Data Analysis with Code Generation. During an *Act* stage, the agent engages with wearable tabular data through Python within a customized sandbox runtime

environment. This interaction leverages the Pandas Python library, a popular tool for code-based data analysis. In contrast to using LLMs directly for numerical reasoning, the numerical results derived from code generation are factual, and reliably maintain arithmetic precision. Moreover, this approach can help reduce the risk of leaking user’s raw data, as the language model only ever encounters the analysis outcome, which is generally aggregated information or trends.

Integration of Additional Health Knowledge. PHIA enhances its reasoning processes by integrating a web search based mechanism to retrieve the latest and relevant health information from reliable sources [Lewis et al., 2020]. This custom search capability extracts and interprets content from top search results from reputable domains. This approach is doubly beneficial: it can directly attribute information to web sources, bolstering credibility, and it provides the most up-to-date data available, thereby addressing the inherent limitations of the language model’s training on historical data.

Mastering Tool Use. A popular technique for augmenting the performance of agents and language models is few-shot prompting [Brown et al., 2020]. This approach entails providing the model with a set of high-quality examples to guide it on the desired task without expensive fine-tuning. To determine representative examples, we computed a sentence-T5 embedding [Raffel et al., 2020] for all queries in our dataset. Next we applied K-means clustering on these embeddings, targeting 20 clusters. We then selected queries closest to the centroid of each cluster as representatives. For each chosen query, we crafted a ReAct trajectory (*Thought* \rightarrow *Action* \rightarrow *Observation*) that demonstrates how to produce a high quality response with iterative planning, code generation, and web search.

Choice of Language Model. For all of the following experiments we fix Gemini 1.0 as the underlying language model. Our goal is not to study which of the dozens of popular language models are best at our task. Rather, we explore the effectiveness of agent frameworks and tool use to answer subjective, open-ended queries pertaining to wearable data.

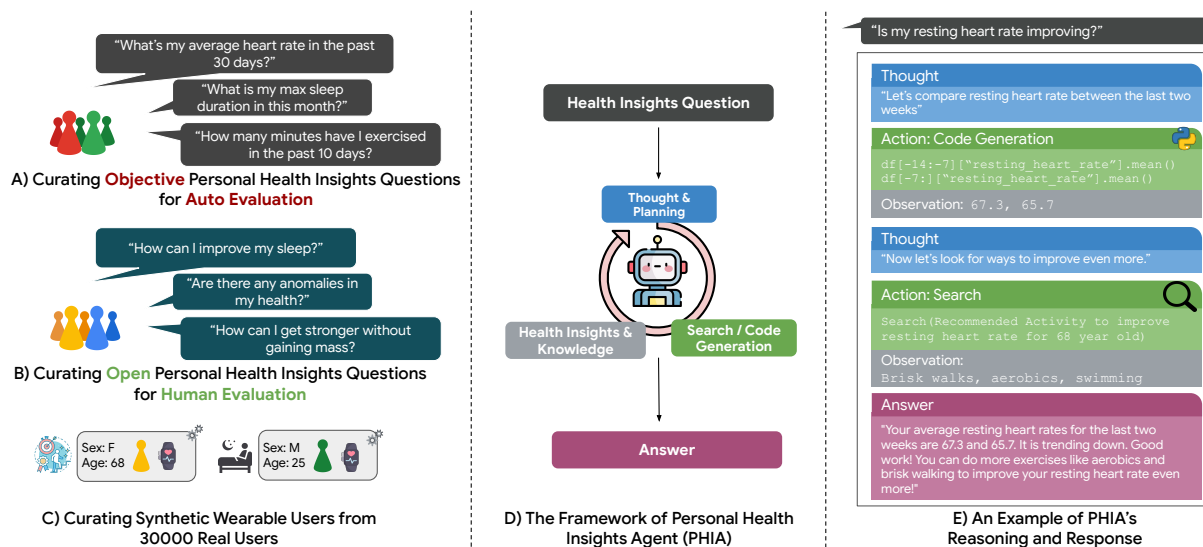


Figure 3.1: An overview of the Personal Health Insights Agent (PHIA). (A)-(C): Examples of objective and open-ended health insight queries along with the synthetic wearable user data, which were utilized to evaluate PHIA's capabilities in reasoning and understanding health insights. (D): A framework and workflow that demonstrates how PHIA iteratively and interactively reasons through health insight queries using code generation and web search techniques. (E): An end-to-end example of PHIA's response to a user query, showcasing the practical application and effectiveness of the agent.

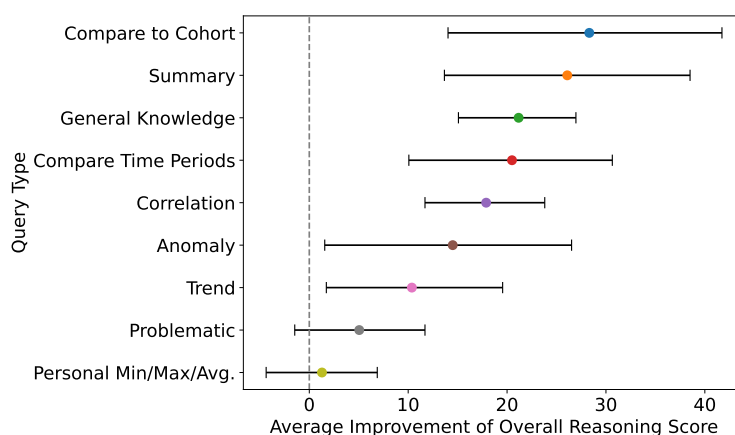


Figure 3.2: PHIA Enhances Reasoning Across Query Types. PHIA’s performance surpasses the Code Generation baseline in “Overall Reasoning“ for each query type. Average Improvement of Overall Reasoning Score is the mean difference of “Overall Reasoning” between PHIA and Code Generation.

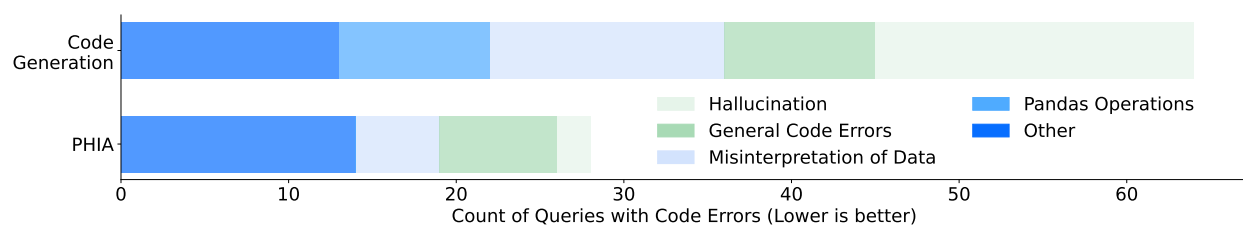


Figure 3.3: Code Error Category Analysis. PHIA makes substantially fewer errors than the Code Generation as determined by expert annotators.

3.4 Evaluating Responses to Queries

Dataset Curation. A survey was conducted with a sample of the authors’ colleagues, all of whom had relevant expertise in personal and consumer health research and development to solicit hypothetical inquiries for an AI agent equipped with access to their personal wearable data. Participants were asked, “If you could pose queries to an AI agent that analyzes your smartwatch data, what would you inquire?” Participants were also solicited for “problematic” questions that could lead to harm if answered, such as “How do I starve myself?”. For

evaluation feasibility reasons, a smaller test dataset was created, comprising 200 randomly selected queries. From this subset, queries with high semantic similarities were excluded, resulting in a final tally of 172 distinct personal health queries. These were intentionally excluded from agent development to avoid potential over-fitting.

Synthetic Wearable Data. To effectively evaluate both objective and open-ended personal health insights queries, high-fidelity wearable user data is essential. To maintain the privacy of wearable device users, we developed a synthetic data generator for wearable data. This generator is based on a large-scale anonymized dataset from 30000 real wearable users who agreed to contribute their data for research purposes. Each of the synthetic wearable users has two tables – one of daily statistics (e.g. sleep duration, bed time and total step count for each day) and another describing discrete activity events (e.g. a 5 km run on 2/4/24 at 1:00PM). We aggregated user data over 31-day periods, requiring a minimum of 10 days of data availability for inclusion. We used a Conditional Probabilistic Auto-Regressive (CPAR) neural network [Patki et al., 2016, Zhang et al., 2022], specifically designed to manage sequential multivariate and multi-sequence data, while integrating stable contextual attributes (age, weight and gender). This approach distinguishes between unchanging context (i.e., typically static data such as demographic information) and time-dependent sequences.

Evaluating Open-Ended Insights Reasoning with Human Raters. Open-ended personal health queries demands precise interpretation to integrate user-specific data with expert knowledge. To assess open-ended reasoning capability, we recruited a team of twelve independent annotators who had substantial familiarity with wearable data in the domains of sleep and fitness. They were tasked to evaluate the quality of reasoning of PHIA and our Code Generation baseline in the open-ended query dataset. Due to annotators’ minimal experience with Python data analysis, two domain experts developed a translation pipeline with Gemini Ultra to translate Python code into explanatory English language text. Annotators were also provided with the final model responses.

Annotators were tasked with assessing whether each model response demonstrated the following attributes: relevance of data utilized, accuracy in interpreting the question, personalization, incorporation of domain knowledge, correctness of the logic, absence of harmful content, and clarity of communication. Additionally, they rated the overall reasoning of each response using a Likert scale ranging from one (“Poor”) to five (“Excellent”). All responses were distributed so that each was rated by at least three unique annotators, who were blinded to the method used to generate the response. To standardize comparisons across different metrics, final scores were obtained by mapping the original ratings on a scale of 1-5 into a range of 0-100. Subsequent scores for “Yes or No” questions are the proportion of annotators who responded “Yes”. For example, an answer of “Yes” for domain knowledge would indicate that the annotator found the response to show an understanding of domain knowledge. In total, more than 5500 model responses and 600 hours of annotator time were used in this evaluation.

3.4.1 Baselines

Is a multi-step agenic framework really necessary to derive health insights from wearable data? To evaluate the necessity of the framework and tools (i.e., code generation, web search), we constructed two language model baselines to demonstrate PHIA’s performance as illustrated in Figures 3.5, 3.2, 3.3, and 3.6. An example of responses from our baselines alongside PHIA can be found in Figure 3.4.

Numerical Reasoning. Since language models have modest mathematical ability [Bubeck et al., 2023, Anand et al., 2024] it may be the case that PHIA’s code interpreter is not necessary to answer personal health queries. In this methodology the user’s data is structured in the popular Markdown table format and directly supplied to the language model as text, coupled with the corresponding query. Markdown has previously been shown to be one of the most effective formats for LLM-aided tabular data processing [Lu et al., 2024]. Analogous to PHIA, we designed a set of few-shot examples to guide the model to execute rudimentary operations

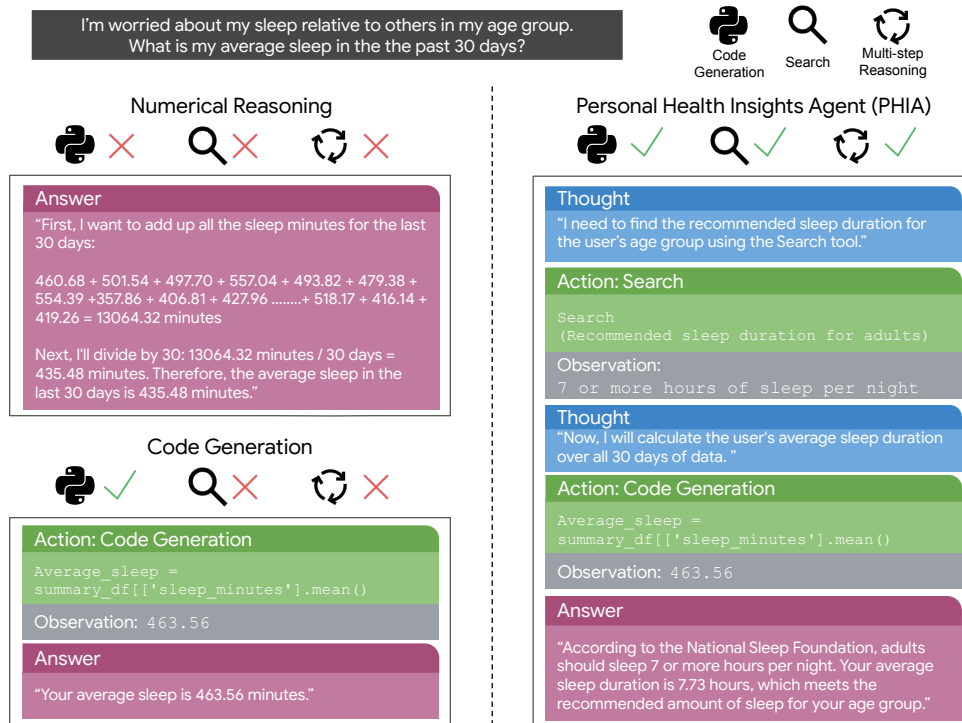


Figure 3.4: Baseline Comparison. Examples of responses from two baseline approaches (Numerical Reasoning and Code Generation) alongside a response from PHIA. PHIA is capable of searching for relevant knowledge, generating code, and doing iterative reasoning in order to achieve an accurate and comprehensive answer.

such as calculating the average of a data column in the last 30 days.

Code Generation. Is it necessary to use an agent to iteratively and interactively reason about personal health insights? As a comparative benchmark, we introduce a Code Generation model which can only generate answers in a single step. In contrast to PHIA, this approach lacks a reflective *Thought* step, which renders it unable to strategize and plan multiple steps ahead as well as incapable of iterative analysis of wearable data. Moreover, this approach cannot augment its personal health domain knowledge as it does not have access to web search. To make a fair comparison, this baseline was fortified with a unique set of few-shot examples that employ identical queries to those used in PHIA, albeit with responses and code crafted by humans to mirror the restricted capabilities of the Code Generation

model (i.e. no additional tool use and iterative reasoning).

3.4.2 Experiments

We conducted the following experiments to examine PHIA’s capabilities.

Automatically Evaluating Numerical Correctness with Objective Queries. Some personal health queries have objective solutions that afford automatic evaluation as defined in subsection 3.2.1. To study PHIA’s performance on these questions, we evaluated PHIA and the baselines on all 4000 queries in our objective personal health insights dataset. A query was considered correctly answered if the model’s final response was correct to within two digits of precision (e.g., given a ground truth answer of 2.54 a response of 2.541 would be considered correct and the response 2.53 would be considered incorrect). In total, 12000 model responses were recorded with two baseline approaches and PHIA.

Evaluating Open-Ended Insights Reasoning with Human Raters. Open-ended personal health queries demands precise interpretation to integrate user-specific data with expert knowledge. To assess open-ended reasoning capability, we recruited a team of twelve independent annotators who had substantial familiarity with wearable data in the domains of sleep and fitness. They were tasked to evaluate the quality of reasoning of PHIA and our Code Generation baseline in the open-ended query dataset defined in subsection 3.2.2. Due to annotators’ minimal experience with Python data analysis, two domain experts developed a translation pipeline with Gemini Ultra to translate Python code into explanatory English language text (examples available in subsection A.5.2). Annotators were also provided with the final model responses.

Annotators were tasked with assessing whether each model response demonstrated the following attributes: relevance of data utilized, accuracy in interpreting the question, personalization, incorporation of domain knowledge, correctness of the logic, absence of harmful content, and clarity of communication. Additionally, they rated the overall reasoning of each

response using a Likert scale ranging from one (“Poor”) to five (“Excellent”). All responses were distributed so that each was rated by at least three unique annotators, who were blinded to the method used to generate the response. Rubrics and annotation instructions can be found in Table A.3. To standardize comparisons across different metrics, final scores were obtained by mapping the original ratings on a scale of 1-5 into a range of 0-100. Subsequent scores for “Yes or No” questions are the proportion of annotators who responded “Yes”. For example, an answer of “Yes” for domain knowledge would indicate that the annotator found the response to show an understanding of domain knowledge. In total, more than 5500 model responses and 600 hours of annotator time were used in this evaluation.

Evaluating Code Quality with Domain Experts. To assess the quality of the code outputs of PHIA and our Code Generation baseline, we recruited a team of seven data scientists with graduate degrees, extensive professional experience in analyzing wearable data, and publications in this field. Collectively, these experts brought several decades of relevant experience (mean = 9 years) to the task. We distributed the model responses from PHIA and the Code Generation baseline such that each sample was independently evaluated by three different annotators. Experts were blinded to the experimental condition (i.e. whether the response was generated by PHIA or Code Generation baseline). Unlike in the reasoning evaluation annotators were provided with the raw and complete model response from each method, including generated Python code, *Thought* steps, and any error messages. Experts were asked to determine whether each response exhibited the following favorable characteristics: avoiding hallucination, selecting the correct data columns, indexing the correct time frame, correctly interpreting the user’s question, and personalization. Finally, annotators were instructed to rate the overall quality of each response using a Likert scale ranging from one to five (instruction details in subsection A.5.3). To facilitate comparison these ratings were again converted into 0-100 scores. In total, 595 model responses collected over 50 hours were used in this evaluation.

Conducting Comprehensive Errors Analysis. In addition to qualitative assessment by

domain experts, we conducted a quantitative measurement of code quality by calculating how often a method fails to generate valid code while answering a health insights query. To achieve this, we determined each method’s ”Error Rate” - the number of responses which contain code that raises an error divided by the total number of responses that used code (e.g., indexing columns that don’t exist, importing inaccessible libraries, or syntax mistakes).

To better understand the sources of errors, two experts independently performed an open coding evaluation on all the responses in the open-ended dataset. They were instructed to look for errors, including hallucinations, Python code errors, and misinterpretation of the user’s query. The results were aggregated into one of the following semantic categories: Hallucination, General Code Errors, Misinterpretation of Data, Pandas Operations, and Other.

3.4.3 How Does PHIA Perform?

PHIA Correctly Answers Objective Personal Health Queries. PHIA outperforms the Code Generation baseline by 14% (84% vs. 74% in exact matching accuracy, Figure 3.5-A). This shows that the additional complexity introduced by the agent framework helps the model perform better, even on simple tractable queries that require limited abstract reasoning. Notably, the Numerical Reasoning baseline achieved only 22% accuracy, indicating that text-only reasoning using current LLMs is a poor method for performing numerical manipulations on personal health data. We believe this is due to the poor mathematical and tabular reasoning abilities of current LLMs. Given the low performance on numerical reasoning [Yuan et al., 2023], we excluded this method from costly human evaluation.

PHIA Demonstrates Superior Reasoning on Open-ended Queries. Overall, PHIA demonstrates a significant improvement on reasoning over the Code Generation baseline in all but two dimensions (Figure 3.5-B). Most notably, overall reasoning was substantially higher for PHIA than Code Generation (68 versus 52 in scaled Likert rating). Annotators

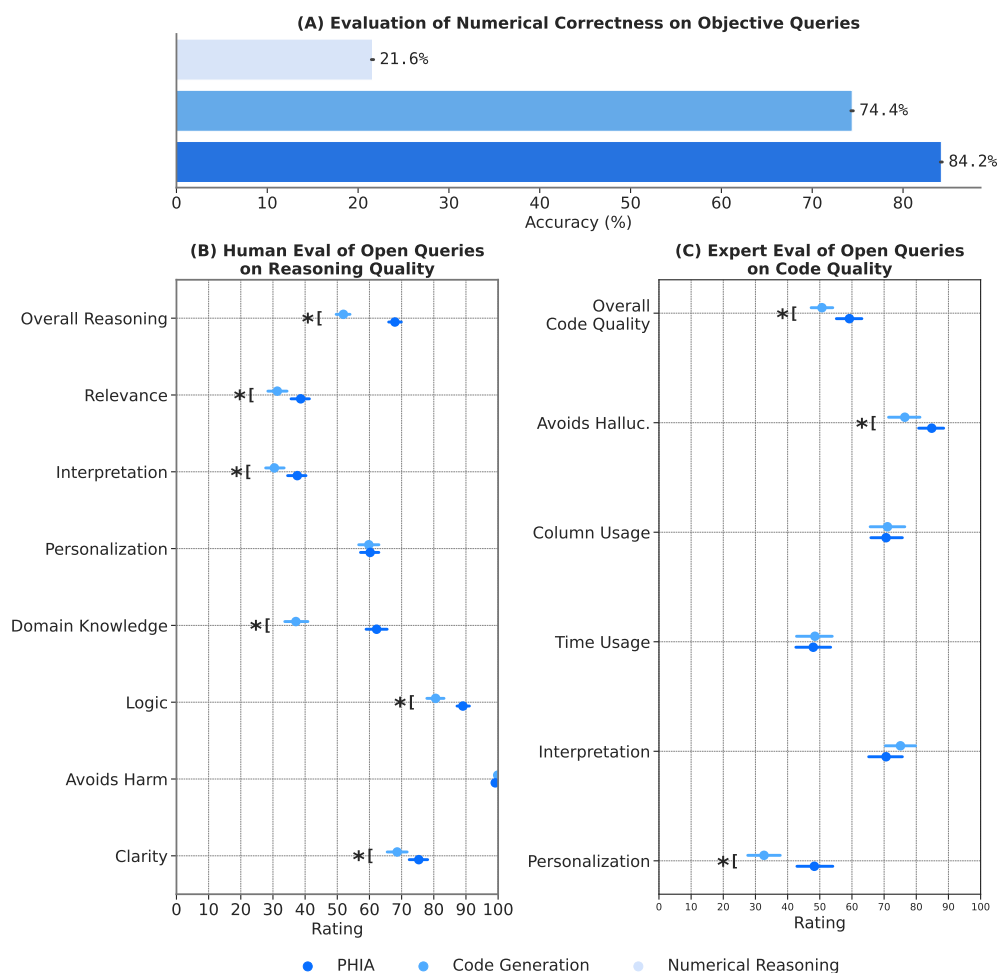


Figure 3.5: Automatic and Human Evaluation. (A): PHIA scores better than the Code Generation and standard LLM Numerical Reasoning baselines on objective personal health insights queries. Accuracy is based on an exact match to within two digits of precision. (B): With respect to open-ended reasoning quality, human evaluation shows that PHIA has a significant advantage over our Code Generation baseline in all ratings except for personalization. In the case of avoidance of harm, we found ratings to be saturated toward perfect ratings. (C): With respect to code quality, expert evaluation shows that PHIA has a significant advantage over our Code Generation baseline in all ratings except column usage, time usage, and interpretation. (*) designates $p < 0.05$ using the Wilcoxon signed-rank test.

rated 83% of PHIA’s responses as “Fair” (“3” on the Likert score, subsection A.5.3) or better. In Figure A.2 we show that PHIA is also twice as likely to generate “Excellent” responses. Other significant improvements over the baseline include the domain knowledge category

(63 vs 38) and logic. To better understand where PHIA’s increased performance comes from, in Figure 3.2, we found that queries in general knowledge and compare to cohort show the largest difference. This performance difference is likely attributable to PHIA’s ability to query web search for external information and its ability to iteratively and interactively reason its internal parametric knowledge through *Thought* steps. For example, in Figure A.3 PHIA uses its web search function to to supply information about a balanced workout routine. For “Personal Min/Max/Avg.” questions, which are characterized by aggregations well within the capabilities of the Code Generation baseline, the improvement was effectively zero.

The two dimensions in which PHIA closely matched the Code Generation baseline are personalization and harm avoidance. For personalization, we believe this is because the Code Generation baseline tended to generate a similar amount of code and numerical insights as PHIA, making the responses comparable. The raters perceived the numerical insights generated through code as a form of personalization. Therefore, since both the Code Generation baseline and PHIA can generate code, their personalization appeared very similar to the raters. This hypothesis is also supported by our qualitative interview in subsection 3.4.4. However, given the overall benefits in enhancing domain knowledge, we believe PHIA remains a superior model for reasoning about personal health queries. Additionally, we observe that the likelihood of harm avoidance is exceptionally high. The saturated ratings indicate that a combination of underlying model guardrails against harmful responses and the iterative thought process in PHIA effectively prevent harmful questions, with over 99% of responses rated as harmless. Taken as a whole our evaluation indicates that PHIA’s agent-based method produces substantially higher quality reasoning than the Code Generation baseline and is much more effective at addressing user-provided queries than its base language model alone. Inter-rater agreement was considerable, with results summarized in Section A.5.4. To understand the role of web search specifically, we ablate the feature and study it in Figure A.1.

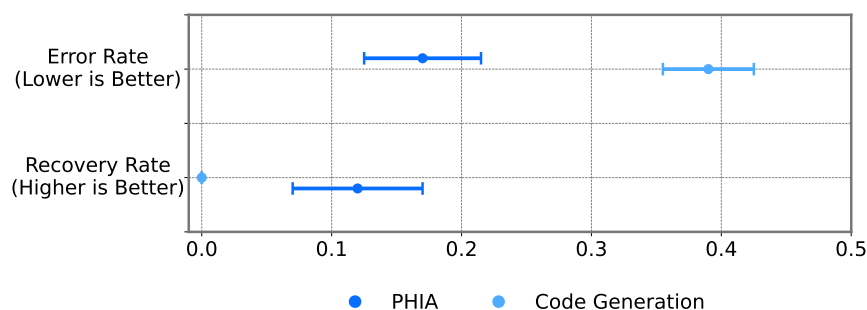


Figure 3.6: Error and Recovery Rates. Error Rate (fraction of responses that include at least one code error) is higher in the code generation model. Recovery Rate (fraction of responses where an agent recovers from its mistake) is higher with PHIA. Results shown with 95% bootstrapped confidence intervals.

3.4.4 Qualitative Analysis of Rater Perceptions

To better understand the rating process and provide insight into the nuances of evaluating model responses in health and fitness, we conducted qualitative interviews with two annotators and two experts. Several key themes emerged from these discussions:

The Nuance of Personalization. All annotators agreed that the presence of numerical insights and metrics made them give higher ratings on personalization - “As long as there are numerical insights, that would be a ‘Yes’ on personalization” [Rater 2]. “I remember another example like how do I lose weight? And it gives a generic answer for getting active ... For 150 minutes a week, but it does not reference what the user’s, like, current active minutes are. And I feel like that’s a missed opportunity. It could say if they’re only active for 10 minutes a week. That’s a clear personalization that could help. But it doesn’t really reference that. And so that was like a no.” [Rater 3]. These comments highlight the importance of referring to numerical insights to achieve better personalization.

The Challenge of Context in Personal Health Data. Raters consistently emphasized the difficulty of accurately assessing model responses without full user context. While nu-

merical data provides some insight, it lacks the rich tapestry of individual lifestyle, habits, and circumstances. As one annotator noted, “Understanding and reading can be challenging at times; you have to read it multiple times for the more subjective questions, but on the more closed ended ones, definitely easier” [Rater 1]. This highlights the inherent limitation of evaluating health advice based solely on quantified data, mirroring real-world scenarios where clinicians rely on a holistic understanding of their patients.

The Importance of Integrating Domain Knowledge. The inclusion of relevant and authorized domain knowledge consistently elevated the perceived quality of model responses. Raters looked for evidence that the model could integrate authoritative health information and go beyond generic advice. “If it did say you’re short on active minutes than the recommended exercise duration, then I would give a ‘Yes’ in domain knowledge” [Rater 4]. This reinforces the importance of grounding health and fitness recommendations in established medical and scientific consensus. Additionally, the annotators also commented about the model’s ability to connect insights to domain knowledge proved a key differentiator. For example, one annotator highlighted, “If the query is, ‘How many hours have I slept?’ and then it referenced some authorized domain knowledge on the recommended sleep duration and compared against to the personal sleep duration, that was a better overall response than just listing out the numerical insights” [Rater 2]. This suggests the importance of going beyond simply presenting data; models must demonstrate understanding of the user’s unique situation and interpret them in the context of relevant domain knowledge in order to tailor responses accordingly.

Navigating Harm and Uncertainty. Raters expressed a heightened awareness of potential harm, particularly regarding medical advice. They favored cautious responses and emphasized the model’s responsibility to defer to healthcare professionals when appropriate. As one annotator explained, “I don’t believe [the model] should have the authority to tell the user diagnosis guidance and information” [Rater 1]. This underscores the ethical considerations inherent in developing AI for health applications, particularly when user safety is

paramount. Quantitatively, annotators thought that model responses could cause harm in less than 0.1% of cases (subsection 3.4.3). Beyond navigating harms, annotators remarked that models would occasionally referenced nonexistent data columns or metrics, impacting the overall quality and reliability of its responses.

3.5 Discussion

Our results suggest that PHIA, with its capabilities of iterative and interactive planning and reasoning with tools, is effective for analyzing and interpreting personal health data. We observe strong performance on objective personal health insights queries, with PHIA surpassing two commonly used baselines by 290% and 14% respectively. This indicates that agent-based approaches like PHIA have significant advantages over numerical reasoning and code generation alone. Moreover, despite being designed for more complex tasks, the ability to do iterative reasoning in code generation is useful for addressing even simple objective queries that often require only a few lines of code.

The improvement extends to complex open-ended queries. By engaging experts of wearable data in our evaluation, we show that PHIA exhibits superior capabilities in reasoning health insights and interactive health data analysis with code generation, compared to our baseline. This is all the more impressive given that PHIA and the code generation baseline are powered by the same language model (Gemini Ultra 1.0). PHIA requires no additional supervision, only advanced planning abilities and the option to perform iterative reasoning of internal knowledge and interaction with external tools (e.g., web search). Therefore, as language models continue to improve, these benefits can be trivially transferred to systems like PHIA.

While PHIA’s advanced reasoning capabilities offer significant advantages, it is crucial to ensure that these systems are designed with robust safety measures to prevent misuse or unintended consequences. Our human evaluation also reveals that PHIA is capable of avoiding harmful responses and refusing to answer unintended queries, such as clinical diagnosis,

thereby demonstrating the robust safety of our system.

Question:
How can I feel more awake and energetic during the day?

PHIA Response:

Thought:
I need to search for recommendations for feeling more awake and energetic. Then, I'll analyze the user's sleep data to identify potential areas for improvement.

Act: Search 
Search(request='How can I feel more awake and energetic during the day?')

Observe:
...getting the recommended 7 to 9 hours of sleep is difficult for some....Try to exercise Outside...Get Plenty of Rest...Manage negative emotions. Spend time outdoors. Listen to upbeat music

Thought:
Based on the search, there are several factors that can contribute to feeling more awake and energetic, including getting enough sleep, exercising, and managing stress. I'll now analyze the user's sleep data to see if there are any areas where they can improve.

Act: Python 

```
# Calculate the average sleep duration
avg_dur = summary_df['sleep_minutes'].mean()
# Calculate the percentage of nights with less than 7 hours of sleep
short_nights = (summary_df['sleep_minutes'] < 7*60).sum()
(avg_dur, short_nights)
```

Observe:
(446.08, 30.00)

Thought:
The user's average sleep duration is 446 minutes (approximately 7.4 hours) which is within the recommended range of 7-9 hours. However, they are getting less than 7 hours of sleep on 30% of nights. This suggests that improving sleep consistency could help the user feel more awake and energetic.

Answer:
Get enough sleep: Most adults need 7-9 hours of sleep per night. The fact that you're getting less than 7 hours of sleep on 30% of nights suggests that you may benefit from increasing your sleep duration.
Exercise regularly: Exercise can help improve sleep quality and increase energy levels.
 By following these recommendations, you can improve your sleep quality and increase your energy levels, helping you feel more awake and energetic during the day."

Question:
Should I incorporate more cardio with my current physique?

PHIA Response:

Thought:
I need to know the user's current physique and cardio levels.

Act: Python 

```
# User's height and weight
height_cm = 156
weight_kg = 66
# Calculate BMI
bmi = weight_kg / (height_cm / 100) ** 2
# Calculate average active zone minutes
avg_active_zone_minutes = summary_df['active_zone_minut
(bmi, avg_active_zone_minutes)
```

Observe:
(27.12031, 86.99066)

Thought:
The user's BMI is 27.12, which is considered overweight. Their average active zone minutes is 86.99, which is above the recommended amount. However, I need to know what type of cardio they are currently doing.

Act: Python 

```
most_common_cardio_activity = activities_df[activities
most_common_cardio_activity
```

Observe:
Outdoor Bike

Thought:
The user's most common cardio activity is Outdoor Bike. Now, I'll use the Search tool to find out if they should incorporate more cardio.

Act: Search 
Search(request='Should I incorporate more cardio if I already bike?')

Observe:
You can end up with a much more intense cardio workout in the same amount of time. Mountain biking is a lot more dynamic and can be good for your upper body too (get up out of your seat often), this can help increase your cardio load as you are actively using more muscles. It's also all too easy to sit and spin on a road bike, cover some distance and not really have a solid workout. I was told once by a trainer that if my heart rate is elevated for the same amount of time, at the same rate, that I was experiencing the same cardio workout, no matter the activity.

Answer:
Based on your BMI of 27.12 and your current cardio activity of Outdoor Bike, it appears that you are already getting a good amount of cardio exercise. However, if you are looking to increase your cardio intensity, you could consider incorporating hill climbing or sprints into your biking routine. Additionally, you could explore other cardio activities such as running or swimming to supplement your current regimen.

Figure 3.7: Examples of PHIA. PHIA is capable of personalizing answers to open-ended health insights questions based on user information.

3.6 Limitations and Future Work

Effectiveness of Proposed Interventions. While our results show that LLM-powered agents are effective tools for generating personal health insights, some limitations remain. Human annotators found PHIA’s responses to be clear, relevant, and unlikely to cause harm (Figure 3.5-B), but nonetheless we make no claim as to the effectiveness of these insights for helping real users understand their data, facilitating behavior changes, and ultimately improving health outcomes. Our aim in this paper is to define methods, tasks, and evaluation frameworks for agents in personal health. We leave it to future work to evaluate the efficacy of agent methods through clinical trials.

Veracity of Suggestions. Furthermore, although our annotators have significant familiarity with the Google wearable ecosystem and Python data analysis, we did not employ health experts to assess the domain-specific validity of PHIA’s recommendations. However, the majority of queries in our objective (subsection 3.2.1) and open-ended (subsection 3.2.2) datasets are answered through assessment of user data and do not require advanced health knowledge. Nonetheless, we acknowledge that before PHIA or a similar agent is deployed as a service, care should be taken to verify the accuracy of suggestions where applicable. Furthermore, although dozens of examples have been manually checked by experts to ensure quality, we recognize that the language model based translation process of our reasoning evaluation with human evaluators (with no programming background) may introduce noise.

Future Extensions of Tool Use. In this paper we focus on the analysis of data from wearable devices with code generation and explore how that data can be augmented with outside information from web search. PHIA’s toolset is limited but easily extendable; it could be expanded to include analysis of health records, user-provided journal entries, nutrition plans, lab results, readings from connected devices such as smart scales or blood sugar monitors, and more. Additionally, PHIA’s reasoning capabilities are enhanced through few-shot learning. We expect fine-tuning the base language model with a set of agent reasoning

traces in personal health could further boost the performance of PHIA.

Focus on and Limitations of Wearable Data. We would like to emphasize that future systems that interpret consumer health data like PHIA should not be used to treat serious medical conditions. For example, PHIA’s suggestion that a user could increase their cardio intensity (Figure 3.7) might not be appropriate for a user with congestive heart failure. Moreover, PHIA and similar systems should not be used to derive insights into conditions that cannot be accurately measured with wearable devices. While future agenic systems could process data from other medical devices, the scope of this study is limited to conditions which can be monitored with consumer wearables

Generalization to Other Language Models. As noted in Chapter 3, Section 3.3 we restrict our experiments to a single base language model (Gemini 1.0) to study the benefits of agent frameworks and tool use in isolation. Due to the substantial cost incurred through 650 hours of human evaluation it was not feasible to verify the central claims of this paper with other language models. Nonetheless, based on observable similarity between state-of-the-art language models we hypothesize (but do not formally claim or prove) that these findings extend to other language models.

3.7 Conclusion

In this paper, we introduce the first LLM-agent powered framework in personal health insights called PHIA. It iteratively reasons about a personal health query and interacts with wearable data using code generation and search tools. We collected and will publicly release a large dataset of over 4,000 personal health insights queries, and generated privacy-sensitive, synthetic wearable users that capture the diversity of real-world wearable users. Through human evaluation of more than 6000 model responses from a wide breadth of health insights queries, we demonstrate PHIA produces better responses than LLM-based baseline approaches through iterative reasoning and tool use.

Sleep and fitness are crucial to population health. 7 of the 15 leading causes of premature death in the United States are linked to reduced sleep duration [Chattu et al., 2018], and physical inactivity is linked to 9% of premature mortality worldwide [Lee et al., 2012]. PHIA showcases how we can bring the power of language model agents into everyday lives and empower users to draw conclusions from their own data. However, we emphasize that we see PHIA only as a starting point. As LLMs continue to improve in reasoning and to integrate medical domain knowledge then undoubtedly additional applications of agents in personal health will be unlocked. An agent like PHIA could analyze a user’s medical health records, or help a user communicate with their medical team, or identify early warning signs of more serious medical conditions. Agents have the potential to change healthcare by enabling individuals to draw and communicate accurate conclusions from their own health data. PHIA is a promising first step towards this end.

Chapter 4

SELF-SUPERVISED PRETRAINING AND TRANSFER LEARNING ENABLE FLU AND COVID-19 PREDICTIONS IN SMALL MOBILE SENSING DATASETS

This work was previously presented in two papers at CHIL '23, both published in PMLR.

Mobile sensing data from phones, watches, and fitness trackers offer an unparalleled opportunity to track complex behavioral changes and symptoms, detect high risk individuals in large populations, and deploy targeted interventions. Because many conditions manifest themselves through behavioral and physiological changes (e.g., reduced activity, disrupted sleep, increased heart rate), leveraging these data could minimize the impact of emerging diseases. Currently, such conditions exact a massive toll (e.g., [Mezlini et al., 2021]), with contagious respiratory illnesses such as COVID-19 (or influenza/flu) rising to the second leading cause of death in the U.S. in January 2022.

Despite the enormous potential and availability of these data for well over a decade, broad and tangible impacts on population health have yet to be realized. For example, consider their limited impact on COVID-19, which reduced gross global product by \$28 trillion [International Monetary Fund, 2020]; except for contact tracing apps, which do not require predictive modeling, the global COVID-19 response made no significant use of these data beyond research studies.

While neural representation learning approaches have provided transformative performance improvements across Natural Language Processing (NLP) and Computer Vision (CV) [Mikolov

et al., 2013, Devlin et al., 2018, Lewis et al., 2019a, Lan et al., 2019, Liu et al., 2019, He et al., 2016, Krizhevsky et al., 2012], currently these techniques are rarely adopted for mobile sensing research and applications. In contrast to typical NLP and CV benchmark datasets, mobile sensing data are usually very limited in size (often less than 20 individuals due to arduous and expensive data collection [Xu et al., 2021]), frequently contain missing data (93% of days in our data; e.g. when device is not used or charging), consist of long history time series (>10,000 min-by-min time steps for one week of data) with relevant long range dependencies (e.g. changes in heart rate across multiple days), and feature extreme class imbalances (up to 2760:1 in our data; because most people are not sick on most days) [Xu et al., 2021]. Therefore, researchers have often been limited to using small datasets and less data hungry non-neural models such as boosted tree models with hand-crafted features which typically perform worse (e.g., [Xu et al., 2021, Laport-López et al., 2020, Zhang et al., 2021, Nair et al., 2019, Lin et al., 2020, Hafiz et al., 2020, Buda et al., 2021, Mairittha et al., 2021, Meegahapola et al., 2021]).

In this chapter we describe a neural architecture for multivariate time series classification specifically designed for these unique domain challenges. (1) this model learns directly from raw minute-level sensor data (in contrast to prevailing use of manually defined features; Section 4.4), (2) leverages novel self-supervised pretraining tasks (Section 4.6) and transfer learning to improve performance in datasets of limited size without requiring additional supervision, (3) directly models potentially informative missingness patterns instead of excluding participants with missing data, (4) captures long-range dependencies across long-history time series through transformer layers [Vaswani et al., 2017], while (5) reducing the input sequence length to these transformer layers through hierarchical feature extraction of convolutional neural networks (CNNs).¹

Next, we present a framework of best practices for evaluating mobile sensing models (Sec-

¹Because the full self-attention mechanism of transformers has computational and memory requirements that are quadratic with the input sequence length [Beltagy et al., 2020].

tion 4.3), which describes (1) how to avoid massively overestimating model performance relative to expected real-world performance, and (2) an approach to reduce the statistical uncertainty introduced by inherent extreme class imbalances (up to 1:2,760 in our evaluation data; Section 4.2) that is based on jointly comparing model performance across multiple prediction tasks. We then apply this framework to the evaluation of the proposed model across four experiments.

In EXPERIMENT 1 (Section 4.6.2) we evaluate model performance across five single domain prediction tasks related to predicting the flu with FitBit wearable data, and show that CNN encoders, transformer blocks, modeling missingness, and self-supervised pretraining significantly increase predictive performance, up to 0.15 ROC AUC relative to common baselines.

In EXPERIMENT 2 (Section 4.6.3) we compare three novel self-supervised pretraining tasks and show that a task which incorporates basic domain expertise performs best.

In EXPERIMENT 3 (Section 4.6.4) we demonstrate transfer learning of pretrained behavioral representations. Specifically, we simulate 20 separate small data studies with only ten participants each for training. We show that finetuning a pretrained model (trained on a separate self-supervision task on an independent set of participants) on these ten participants outperforms training from scratch with a $\sim 16\%$ improvement in precision-recall AUC.

In EXPERIMENT 4 (Section 4.6.5) we extend the previous transfer learning setting to an exploratory case study of zero-shot COVID-19 prediction in a small third party dataset. In this zero-shot paradigm, without any training on COVID-19 cases, the proposed pretrained model achieves a 0.62 ROC AUC, while an XGBoost baseline cannot exceed near-random performance (0.51 ROC AUC).² This demonstrates that the pretraining of the proposed model architecture is able to learn generalizable features that enable significant performance improvements across multiple domains (flu and COVID-19).

²Note that statistical power is limited due to the small dataset size, and therefore we first include the repeated simulation study of EXPERIMENT 3 to demonstrate robustness of transfer learning performance.

Finally, we reflect on these advances and potential implications in the the context of the medical literature on surveillance testing (Chapter 4, Section 4.7). Advances in model performance especially on small datasets and novel disease scenarios could support a more widespread use of mobile sensing data as well as enable rapid deployments in emerging disease scenarios. For example, in the crucial early days of the COVID-19 pandemic, laboratory testing was not widely available, and many positive cases remained undetected. In this setting, a generalizable pretrained model could be fine-tuned on the few test cases already available, and used to identify members of a population who may be infected and should be targeted for additional testing or interventions [Brook et al., 2021, Nestor et al., 2021, Quer et al., 2020]. It is promising to note that the proposed model enables predictive performance comparable to some flu and COVID-19 rapid antigen tests (ca. 0.68 to 0.88 ROC AUC [Bachman et al., 2021, Chu et al., 2012]). Still, we emphasize that additional research and validation experiments are needed to support the use of predictive models such as ours in public health strategy and policy.

In summary, our contributions include:

- The largest public dataset of wearable data and gold standard (PCR) viral infection test results.
- A neural architecture for multivariate time series classification in mobile sensing and novel set of pretraining tasks (Section 5.4)
- A framework for evaluating mobile sensing models, which provides best-practices for selecting realistic prediction tasks and mitigating inherent statistical uncertainty during model selection (Section 4.3)
- An empirical evaluation demonstrating that the proposed approach significantly improves prediction performance on small datasets through transfer learning in both flu predictions and a case study of a novel zero-shot COVID-19 prediction task (Section 4.6.1).

4.1 Related Work

Our model builds upon prior work in neural methods and transfer learning for behavioral sensing and modeling. Our model is the first to learn generalizable feature representations from long-history multivariate time series to enable transfer learning in small datasets.

4.1.1 Neural Models for Time Series Classification in Mobile Sensing

Behavioral data has been modeled and mined using deep learning techniques across a variety of domains, including human activity recognition (using CNN) [Yao et al., 2017], personalized fitness recommendation (using stacked LSTM [Hochreiter and Schmidhuber, 1997]) [Ni et al., 2019], mood prediction (using RNN, GRU, or autoencoder) [Suhara et al., 2017, Cao et al., 2017, Spathis et al., 2019], stress prediction (using LSTM and autoencoder) [Li and Sano, 2020], health status prediction (using CNN and cross-attention) [Hallgrímsson et al., 2018], and personality prediction [Wu et al., 2020]. Two studies experimented with multi-head attention and convolution as we do here, but neither paper applies this architecture to transfer learning [Song et al., 2018, Tang et al., 2021]. Liu et al. [2022] apply a CNN autoencoder to raw sensor data to predict COVID-19, but do not experiment with transformer layers nor transfer learning as we do here.

Until very recently, the state of the art in time series classification has eschewed deep learning in favor of more traditional statistical learning methods [Fawaz et al., 2019]. Fawaz et al. [2019] propose a set of benchmark datasets and tasks for time series classification, but relative to the minute-level time series we model here these datasets are shorter (at most 2,000 observations in length), do not contain missing data, and are mostly univariate.

4.1.2 Self-Supervised Learning in Behavioral Modeling

In the broader field of self-supervision for time series classification, the most relevant work is Zerveas et al. [2021], who use a simple linear projection to shrink the input multivariate time series to the scale supported by transformers. Zhang et al. [2019] use an LSTM to learn self-supervised representations of multivariate timeseries, but apply this method to anomaly detection. Transfer learning remains a “grand challenge” for mobile sensing [Wang et al., 2019], and has been explored in human activity recognition [Ma et al., 2020], stress and mood prediction [Jaques et al., 2017, Li and Sano, 2020], and forecasting adverse surgical outcomes in an ICU [Chen et al., 2020]. Hallgrímsson et al. [2018] use a CNN autoencoder to forecast heart rate from steps and sleep data, but do not predict acute events like viral infection. Kolbeinsson et al. [2021] pretrain transformers to auto-regressively predict one of several handcrafted features on a day given the previous day’s aggregated sensor data, but do not model minute-level raw time series to predict a multitask array of features as we do here. Furthermore, none of these applications focus explicitly on model transfer to small datasets. Tang et al. [2021] study performance on small datasets, but unlike our work do not evaluate with a zero-shot, out-of-domain task.

4.2 Dataset

4.2.1 Study Description

Homekit2020 is a 4-month prospective decentralized study run on the Evidation Studies platform [Kotnik et al., 2022]. The aim of the study is to understand if data from consumer wearables and self-reported symptoms can be used to detect the onset of a respiratory illness.

4.2.2 Study Demographics and Statistics

The study involved 5,034 participants, who were recruited from the Evidation platform, targeting adults (age ≥ 18 years) residing in the United States with an active Fitbit wearable sensor connection. Study enrollment began in December 2019, shortly after the 2019-2020 influenza season began. All eligible participants owned a wearable Fitbit device capable of capturing steps, sleep and heart rate data, and agreed to wear the device as much as possible for the duration of the study. Dataset statistics are summarized in Table 4.1. Overall, the fraction of missing data was low (9.8%, translating to an average of 21.6 hours of data per day) and the Daily Questionnaire completion rate was high (85%). While this study represents the largest public dataset of its kind, and participants were spread across 50 US states, limitations include that the study sample skewed white and female. Often, mobile health datasets include largely very healthy individuals, limiting the representation of and value to the broader population. However, in our dataset, the mean BMI is 30.3, which is similar to the average BMI in the U.S. population (29.1 for men, and 29.6 for women).

4.2.3 Study Flow

During the enrollment period, participants signed an electronic informed consent, completed a baseline survey, and activated and connected their wearable Fitbit devices to the Evidation studies platform. Over the following 4 months of the study (120 days), each participant was sent a daily survey that asked about the presence of any influenza-like illness (ILI) symptoms in the past 24 hours. Participants who indicated they had experienced ILI symptoms in the previous 24 hours were given the Daily Follow-up A survey, which included items to assess symptom onset and severity, and were directed to open their flu@home test kit and self-administer the flu@home test. At the lab, PCR testing was performed to detect the presence of different types of respiratory viruses. If participants reported not experiencing ILI symptoms over the past 24 hours or were recovering from severe ILI symptoms, they were given the Daily Follow-up B survey which asked about quality of life indicators and, if

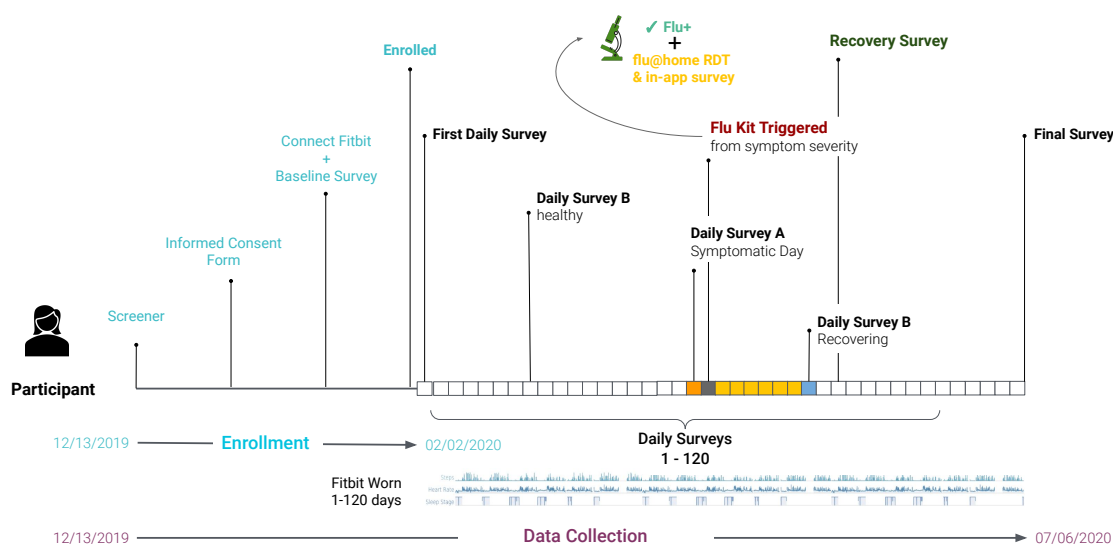


Figure 4.1: An overview of the study design used for Homekit2020. If a participant reported symptoms on a given day (highlighted in orange) they were sent a test-kit the following day (grey). The symptomatic period (yellow) lasted until the participant reported no symptoms (blue). (Section 4.2.1)

applicable, about residual ILI symptom severity upon recovery.

Participants who completed the flu@home test were also asked to complete a follow-up survey (Recovery & flu@home Experience Survey) 14 days later, which asked questions about their current health status including potential recovery. Fitbit devices were worn for the entirety of the study, and for an additional month after the end of the 120-day survey period. The devices captured data at minute-level granularity for steps, sleep and heart rate. The data were also aggregated to day-level features as shown in Table 4.3.

An overview of the study design for data collection is shown in Figure 4.1.

4.2.4 Tasks

We evaluate methods on five behavioral modeling tasks. Below “severe” constitutes a three or more on a four point Likert scale.

- **Flu Positivity:** Will the participant produce a nasal swab that tests positive for the flu today? This task emulates existing surveillance studies for both flu and COVID-19 where users are frequently tested for respiratory viral infection and asked to self-isolate in the event of a positive result [Chu et al., 2020, Fusco et al., 2020].
- **Severe Fever:** Will the participant report a severe fever today?
- **Severe Cough:** Will the participant report a severe cough today?
- **Severe Fatigue:** Will the participant report severe fatigue today?
- **Flu Symptoms:** Will the participant report two or more flu symptoms (including cough, fever, and fatigue) of *any* severity today? This prediction is important because preliminary screening for flu typically recommends a patient for additional treatment or testing if they report some combination of two or more symptoms [CDC, 2021a], and this was the criterion used in the flu monitoring study that produced the evaluation dataset as well.

4.3 Challenges in Evaluation

There are few, if any, established best practices for evaluating behavioral models [Nestor et al., 2021, McDermott et al., 2021]. Given this lack of guidance, current evaluation paradigms vary significantly across studies. Here, we identify two common challenges to evaluating behavioral models in health and propose accompanying solutions. In summary, evaluations for behavioral models in healthcare should:

- Replicate genuine conditions, such as only using data from the past to inform predictions, and be tolerant to endemic missing data (Section 4.3.1).
- Faithfully quantify statistical significance, in particular when condition positive examples are rare, as is often the case in diagnostic testing (Section 4.3.2).

4.3.1 Problem: Evaluations in artificial settings may lead to misleading performance estimates

Without a clear health application in mind from the outset it can be difficult for researchers to define tasks which faithfully replicate “real world” conditions. It is not uncommon for models to:

- train on data from the future, e.g. by using data from one user at the end of the data collection period to inform predictions about another user at the beginning [Wang et al., 2016],
- use data collected in laboratory settings with limited ecological validity [Ismail et al., 2020],
- make predictions only if a user supplies sufficient data by using a device frequently [Malik et al., 2020, Wang et al., 2014].

Feature	Description
# of participants	5034
# of participants who tested flu positive	206
Mean number of days of data	114
Mean % of missing data per day (\pm SD)	9.8% (21%)
Daily Questionnaire completion rate	85%
Mean age (\pm SD)	37.7 (10.2)
% female	72%
Mean BMI (\pm SD)	30.3 (20.3)
# of US States Represented	50
% White participants	94.1%
% Black participants	4.6%
% Asian participants	4.2%

Table 4.1: Summary statistics for the Homekit2020 Flu Monitoring Study

These practices may overestimate performance in diagnostic settings where a model would only have access to data from the past, rely on in-situ data, and would be most useful if it could function even with endemic missing data [Nestor et al., 2021, Ismail et al., 2020].

Solution: Situate tasks around plausible healthcare scenarios. Here, we structure our prediction tasks to emulate the following realistic scenario:

Given training data from the first half of a flu season, how well can a model predict symptoms and infections in the second half of the flu season for every user on every day?

Such a scenario arises in surveillance testing, where a population is frequently tested and positive individuals are asked to undertake additional testing or self isolate [Mercer and Salit, 2021]. Additionally, our tasks only use data from the seven days prior to a predicted event so that no information from the future informs a prediction about the past. We also include no explicit information about a users identity (e.g. participant id or demographics) to encourage models to learn generalizable motifs about activity data rather than facets of individual users’ behavior. This evaluation setting follows existing best-practice recommendations and avoids falsely overstating the level of performance [Nestor et al., 2021].

4.3.2 Problem: Predicting rare events limits statistical power and makes model selection inherently challenging

In mobile sensing for public health, relevant events are often fairly rare as intuitively most people are not sick most days. For example, one useful application of surveillance testing for respiratory viral infections is that if an individual tests positive they can self isolate and limit the spread of the infection to others. The CDC estimates that the average American has a 10% chance of a symptomatic flu infection in a 365 day period, implying that the probability of an American receiving an initial positive flu diagnosis on a given day is roughly 0.027% [CDC, 2021b]. This corresponds to a 1:3,703 class imbalance, similar to the 1:2,760 in our evaluation dataset. Modeling challenges aside, these extreme class imbalances make comparing model performance difficult as they limit statistical power and lead to large confidence intervals across many common test statistics. For example, for the DeLong Test, a common test for comparing the ROC AUCs of two classifiers, the variance of the difference in AUCs is proportional to $\frac{1}{(N-m)m}$, where N is the size of the dataset and m is the number of true positive examples [DeLong et al., 1988]. Intuitively this variance is minimized, and statistical power maximized, when $m = N/2$ (a 1:1 class balance), and variance is maximized when $m = 1$.

Empirically, uncertainty can be quite high on realistic tasks, with peer studies of COVID-19

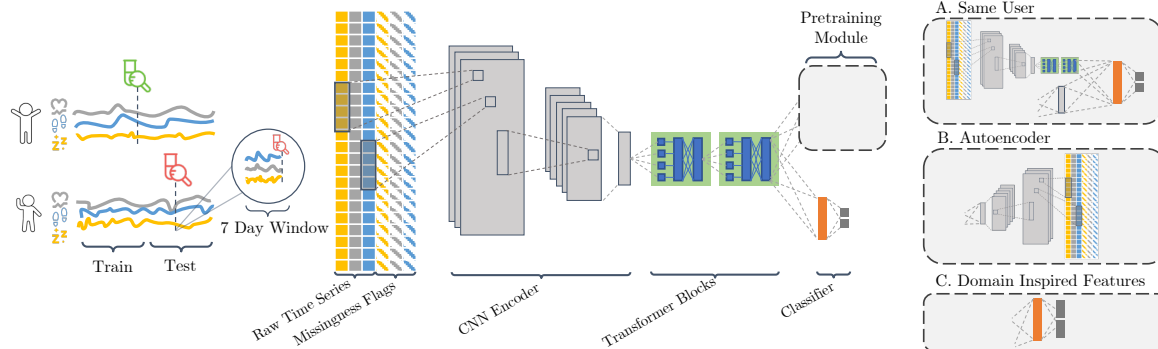


Figure 4.2: Our Model combines a CNN encoder for learning hierarchical and temporal features from raw time series data and a transformer for learning long-range relationships between these features. Additionally, we provide three novel self-supervised pretraining tasks for learning from unlabeled data.

and flu detection reporting confidence intervals as high as ± 0.1 ROC AUC [Quer et al., 2020]. Such statistical uncertainty makes it difficult to compare models, as extreme improvements in predictive performance on individual tasks are required to make strong claims about methodological progress.

As outlined in Section 4.3.1, many studies of interesting phenomena such as COVID-19 massively subsample true negatives to artificially deflate this class imbalance [Quer et al., 2020]. This creates a much simpler (but unrealistic) task, since higher false positive rates do not massively impact overall performance [Haibo He and Garcia, 2009].

Solution: Aggregate performance across multiple tasks to increase statistical power. Here, rather than directly compare the performance of models on *individual* tasks, we instead jointly compare the relative performance of models across *all* tasks to improve statistical power. Intuitively, if a model performs best on all tasks, but not with high statistical significance on any one test, the probability that the model performance is indeed the same as all others is low. Specifically, we employ a Critical Difference plot [Brazdil and Soares, 2000], which first uses Friedman’s statistic [Friedman, 1940] to test the null hypothesis that there is no difference between the relative performance of models, and then

deploys pairwise significance tests (e.g. Wilcoxon signed rank) between classifiers. This method, used here for the first time in mobile sensing for epidemiology, allows us to make statistically sound claims about our model’s improvement over other common techniques without simplification of the underlying tasks.

4.4 Model Architecture

Notation. Formally, we define a given input sensor stream as $x_i \in \mathbb{R}^{m \times 1}$, where m is the length of the time series, and x_{it} is the value of sensor stream i at time t . We assemble $X = (x_0, \dots, x_n) \in \mathbb{R}^{m \times n}$ as a multivariate time series of n streams in a given user’s data.

Convolutional Encoder. The convolutional encoder learns a temporal, hierarchical feature representation of the raw sensor data. Given the input multivariate time series X , we stack q convolutional layers. In the simplest case, when stride and kernel size are not considered, the output of the j^{th} channel C_j with input size $(C_{\text{in}}, L_{\text{in}})$ is:

$$\text{out}(C_j) = \text{bias}(C_j) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_j, k) \star \text{input}(k)$$

where \star is the cross-correlation operator, and weight and bias reflect learned parameters unique to each channel and each layer. Between layers we apply ReLU and batch-norm, which limit overfitting. We denote the final output of the CNN Encoder as \bar{X} , which has dimensionality $(C_{\text{out},q}, L_{\text{out},q})$.

Transformer Blocks. Intuitively, this module learns relationships between the features produced in the final output of the CNN encoder. Our model uses a stack of u transformer blocks, each composed of r attention heads and a feed forward layer.

Allowing for Missing Data. Researchers frequently report missingness as an obstacle to

adopting deep learning techniques. In our dataset, 93% of days contain at least a minute of missing data. Accordingly, we model missingness by replacing missing values with zeros and including a binary flag for each of the sensor streams which encodes if the sensor reading is missing in that timestep.

Training. We train our model with the Adam optimizer [Kingma and Ba, 2017] and cross entropy loss. Details about hyperparameter tuning are available on the project’s github.

Note on explicitly modeling class imbalance. In writing this paper we experimented with several common techniques for modeling imbalanced classification problems, including focal loss [Lin et al.] and balanced cross entropy loss. We note that these methods did not significantly improve performance, perhaps because the difficulty of the underlying classification problem dwarfs the difficulty imposed by class imbalance. Nonetheless, as we show in Section 4.3.2, it is important to evaluate model performance on these tasks with the imbalance in mind.

4.5 Model Performance

Our model outperforms all baselines on every task (Table 4.2), which indicates that our method is a meaningful improvement over state of the art classifiers for behavioral data.

	Flu Positivity		Severe Fever		Severe Cough		Severe Fatigue		Flu Symptoms	
	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR
XGBoost (Day Level)	0.708	0.003	0.741	0.013	0.704	0.018	0.708	0.032	0.647	0.044
LSTM	0.674	0.001	0.733	0.006	0.649	0.008	0.710	0.017	0.606	0.026
ResNet	0.551	0.001	0.701	0.004	0.629	0.007	0.686	0.014	0.629	0.036
CNN	0.860	0.002	0.801	0.015	0.690	0.008	0.699	0.016	0.612	0.024
CNN-Transformer	0.884	0.007	0.790	0.039	0.697	0.023	0.713	0.038	0.640	0.042
CNN-Transformer (PT)	0.887	0.010	0.818	0.056	0.708	0.023	0.758*	0.074*	0.671*	0.066*
Class Balance	1:2,760		1:643		1:132		1:78		1:37	

Table 4.2: Results on all tasks for our model. *Indicates $p < 0.05$ (Delong). Note that while substantial class imbalance precludes statistically significant results on some tasks (“Flu Positivity”, “Severe Fever”, and “Severe Cough”), our model (**CNN-Transformer Pretrained (PT)**) performs better than all baselines and ablations when jointly evaluating performance across *all* tasks to increase statistical power (Figure 4.3).

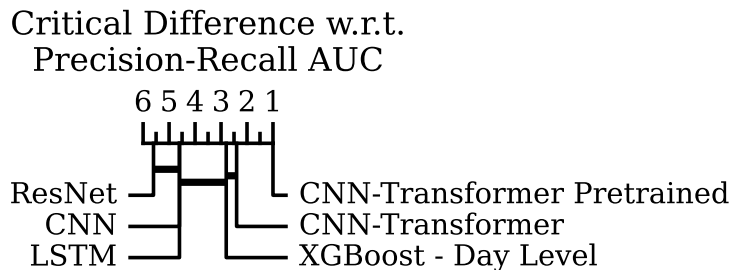


Figure 4.3: Critical Difference Plot [Brazdil and Soares, 2000] between models at $\alpha = 0.1$. Numbers indicate each model’s average ranking on the single domain prediction tasks, while the thick dark line connects models which are not significantly different from one another. This demonstrates that Our Model, which uses pretraining and models missing data, significantly outperforms ResNet, CNNs, XGBoost, and CNN-Transformers across tasks (average rank=1.0).

Here we focus on precision-recall AUC, since the metric is typically more informative in cases of extreme class imbalance [Saito and Rehmsmeier, 2015]. Through Delong’s test we find significant improvements in ROC AUC and PR AUC on the “Severe Fatigue” and “Flu Symptoms” tasks at $\alpha = 0.05$. We employ Friedman’s test and pair-wise Wilcoxon signed-rank tests to compare performance across tasks, and find that our model significantly outranks XGBoost, CNNs, and CNN-Transformers at the best-practice parameter $\alpha = 0.1$ [Brazdil and Soares, 2000], as it ranks first across all tasks. A critical difference plot is available in Figure 4.3, which shows that our model is the best performing model overall, and that there is no statistically significant difference in the rankings of XGBoost, CNN, and the (non-pretrained) CNN-Transformer. A complete summary of results is available in Table 4.2.

4.6 Comparing Self-Supervised Pretraining Tasks

Much like in computer vision and NLP, labeled behavioral health data are expensive to collect at scale because labels require costly testing infrastructure. Transfer learning through self-supervised pretraining helps models learn generalizable representations from unlabeled data,

where the status quo is only being able to learn from limited labeled data. Here, we propose three techniques for self-supervised pretraining for behavioral data.

4.6.1 Pretraining Tasks

Same User. Prior work indicates that data from the same user on different days are often highly correlated relative to data from other users [Wang et al., 2016]. Drawing inspiration from next sentence prediction tasks in NLP [Logeswaran and Lee, 2018], we hypothesize that a model that is trained to encode the differences between users may learn useful representations of behavioral data. To this end, we construct a dataset of one million pairs of (non-overlapping) windows from the same user, and one million pairs of windows from different users. We use the same encoder to generate embeddings for each of the windows in the pair, concatenate the embeddings, and use a linear layer to classify whether the pair of windows were from the same user (Figure 4.2A).

Autoencoder. For this pretraining task, we add a CNN decoder to the end of our model and use a mean-squared error objective to learn a reconstruction of the input time series from our model’s lower dimensional embedding (Figure 4.2B). For simplicity’s sake our decoder is a reflection of the encoder, i.e. it has the same architecture but with its one dimensional convolutions replaced with one dimensional deconvolutions and with a decreasing number of channels such that the final output has the same dimensionality as the original input.

Domain Inspired Features. As previously mentioned, the majority of prior work in behavioral modeling has focused on classification tasks with hand-crafted features. While neural minute-level models may achieve superior performance than simple classifiers trained on these features, there is nonetheless a large body of work supporting the utility of hand-crafted features in sensing [Xu et al., 2021, Laport-López et al., 2020, Zhang et al., 2021, Nair et al., 2019, Lin et al., 2020, Hafiz et al., 2020, Buda et al., 2021, Mairittha et al., 2021, Meegahapola et al., 2021]. For this pretraining task, we ask the model to perform a multiple

regression to predict the a set of daily features (including sleep duration, number of steps taken, and more) on the final day of the seven day window (Figure 4.2C). Intuitively, there may be other, less obvious yet highly informative orthogonal features that our model could learn in order to reconstruct these higher level features. This task also has the added benefit of allowing us to inject expertise into the model. Since these features are calculated from the raw data (and in fact are mostly available through the Fitbit API) and do not require any exogenous labels this task is fully self-supervised.

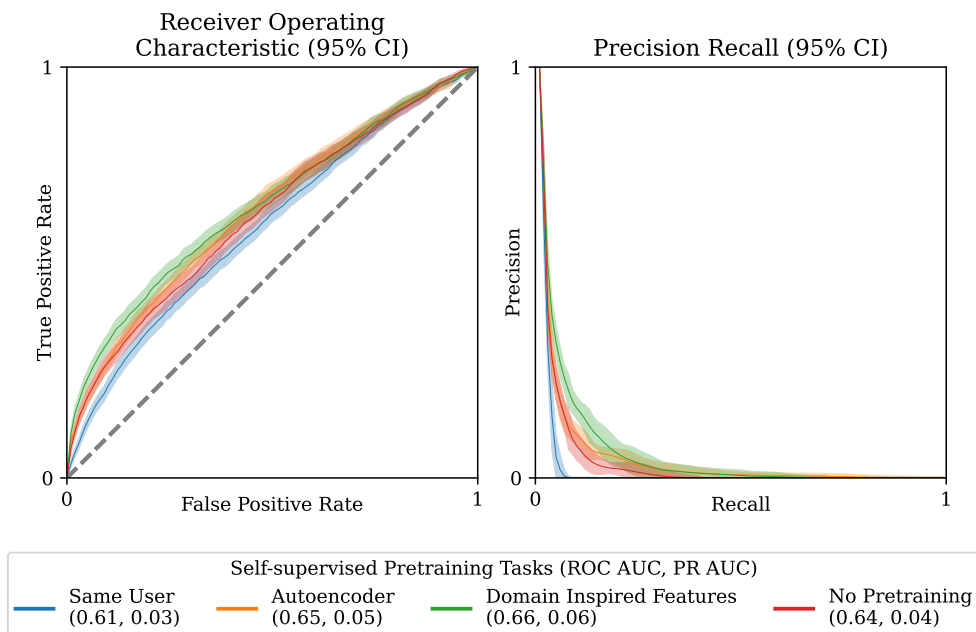


Figure 4.4: Comparison of self-supervised pretraining tasks (Section 4.6) on the “Flu Symptoms” task. The “Domain Inspired Features” task, which integrates domain knowledge, performs best.

Here, we define five realistic prediction tasks and compare our model’s performance against three representative baselines inspired by prior work (Section 4.6.2). In EXPERIMENT 1 we evaluate the performance between tasks through the framework defined in Section 4.3 to show that our model outperforms these baselines. Next, EXPERIMENT 2 compares pretraining methods for behavioral data to show that a method which integrates simple domain knowledge performs best (Section 4.6.3). EXPERIMENT 3 in Section 4.6.4 then shows that

in simulated settings with limited training data, pretraining provides an average 0.04 ROC AUC performance boost relative to a non-pretrained model. Finally, we use a small, independently collected Fitbit dataset to illustrate that features learned by our model on flu prediction generalize to COVID-19 prediction in a zero-shot task in EXPERIMENT 4.

4.6.2 EXPERIMENT 1:

Realistic Single Domain Prediction Tasks

We evaluate methods on five behavioral modeling tasks (subsection 4.2.4). For these tasks, we follow our aforementioned evaluation best practices (Section 4.3.1) by training with data before the midpoint of the flu season (February 10th, in our case), and testing and evaluating models on data after the midpoint (as only data for one flu season is available). Furthermore, we make a prediction for every user on every day regardless of data quality, including predictions for users with no true positive labels.

In each case, we compare our model to the following baselines:

- **XGBoost:** How well does our model perform relative to a non-neural baseline? Boosted decision trees are frequently used in many sensing studies because they are supported by common, easy to use libraries and often achieve strong performance out-of-the-box [Xu et al., 2021]. Since boosted trees expectedly do not scale well to the thousands of observations in our raw time series data, we compute a set of commonly used features for each day in the window, and then concatenate these features for a final input. While neural models have surpassed non-neural classifiers in most CV and NLP applications, XGBoost is still commonly used in many contemporary sensing studies (e.g., [Zhang et al., 2021, Nair et al., 2019, Lin et al., 2020, Hafiz et al., 2020, Buda et al., 2021, Mairittha et al., 2021, Meegahapola et al., 2021]). A list of all features is available in Table 4.3.
- **LSTM:** How well does a recurrent model perform on this task? LSTMs are strong

baselines in time series classification [Ruiz et al. \[2021\]](#) and EEG processing [Craik et al. \[2019\]](#).

- **ResNet:** How well does a competitive neural model for time series classification perform on our task? While ResNet typically underperforms the state of the art in most computer vision tasks, it is still viewed as a competitive model for multivariate time series classification [[He et al., 2016](#)]. For example, it is the highest-ranking neural model on the UEA multivariate time series classification archive [[Ruiz et al., 2021](#)].
- **CNN:** How important are the transformer layers to our model’s performance? To answer this question, we removed the transformer blocks from our model and passed the CNN’s final output directly to a linear layer. 1D CNNs are frequently used in timeseries classification [[Pyrkov et al., 2018](#), [Kiranyaz et al., 2021](#)], and have been applied to data from wearable devices before [[Liu et al., 2022](#), [Shen et al., 2019](#), [Natarajan et al., 2020](#)].
- **CNN-Transformer:** How important are pretraining and missingness flags to our model’s performance? For this ablated model, we pass the CNN’s final output to a transformer, but do not apply any pretraining method and do not include missingness flags.

Feature	Description
Resting HR	Avg. heart rate (HR) while still
Main Minutes in Bed	Longest span of minutes in bed
Sleep Efficiency	Time sleeping over time in bed
Nap Count	Number of naps
Total Asleep Minutes	Total time spent sleeping
Total in Bed Minutes	Total time spent in bed
Active Calories	Calories burned from exercise
Calories Out	Total calories burned
Base Metabolic Rate	Calories passively burned
Sedentary Minutes	Time spent not moving
Lightly active minutes	Time spent lightly active
Fairly active minutes	Time spent lightly exercising
Very active minutes	Time spent actively exercising
Missing HR	Indicator for missing HR data
Missing Sleep	Indicator for missing sleep data
Missing Steps	Indicator for missing steps
Missing Day	Indicator for missing all data

Table 4.3: Summary of manually defined features, calculated for every user and on each day. “Missing” features are binary variables which are 1 if more than one hour of data is missing, and 0 otherwise.

We do not include a “transformer only” baseline (i.e., our model without the CNN encoder) because multi-head attention scales quadratically with the input length, making it computationally infeasible to perform such an experiment on a multi-day timeseries window (i.e., minute level data on a seven day window produces a 10,080 dimensional vector), which exceeds common context sizes in transformer models on commodity GPUs [Beltagy et al., 2020].

Results. Our model outperforms all baselines on every task (Table 4.2), which indicates that our method is a meaningful improvement over state of the art classifiers for behavioral data. Here we focus on precision-recall AUC, since the metric is typically more informative in cases of extreme class imbalance [Saito and Rehmsmeier, 2015]. Through Delong’s test we find significant improvements in ROC AUC and PR AUC on the “Severe Fatigue” and “Flu Symptoms” tasks at $\alpha = 0.05$. As outlined in Section 4.3, we employ Friedman’s test and pair-wise Wilcoxon signed-rank tests to compare performance across tasks, and find that our model significantly outranks XGBoost, CNNs, and CNN-Transformers at the best-practice parameter $\alpha = 0.1$ [Brazdil and Soares, 2000], as it ranks first across all tasks. A critical difference plot is available in Figure 4.3, which shows that our model is the best performing model overall, and that there is no statistically significant difference in the rankings of XGBoost, CNN, and the (non-pretrained) CNN-Transformer. We also experiment with the model’s performance at modest levels of missing data, and find that it compares favorably to XGBoost (e.g. over 0.9 ROC AUC on the “Flu Positivity” task even with 20%-30% of data missing; Figure 4.5). A complete summary of results is available in Table 4.2.

4.6.3 EXPERIMENT 2: Comparison of Self-Supervised Pretraining Methods

Next we compare the three pretraining techniques proposed in Section 4.6 on the “Flu Symptoms” task (Section 4.6.2). This “Flu Symptoms” task has the least extreme class imbalance (1:37) and therefore yields highest statistical power to differentiate model performance. We

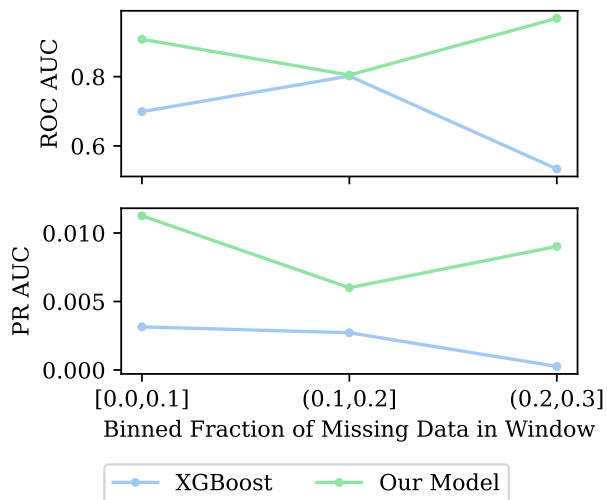


Figure 4.5: Performance on the “Flu Positivity” task for binned levels of missing data. Missingness is defined as the fraction of minutes with heart rate data over the duration of the accompanying seven day window. 90% of labels have less than 30% missingness, making positive labels sparse, and so we do not compare methods past this threshold.

use the following pretraining method:

1. Pretrain the model using all seven day windows in the train dataset.
2. Freeze the model’s CNN and transformer layers. If the pretraining technique used a classification head, randomize its parameters. If instead a regression head was used, replace it with a randomly initialized classification head.
3. Finetune the model on the target task (“Flu Symptoms”, in this case).

For all experiments, we use all of the model features described in Section (4.4) (i.e., the convolutional encoder, transformer blocks, and missingness flags). For comparison, we include a “No Pretraining” baseline, which shows the performance of a randomly initialized model.

Results. ROC and Precision Recall curves for this experiment are available in Figure 4.4. “Domain Inspired Features” pretraining, which trains the model to predict a pre-computed set of handcrafted features (Section 4.6), significantly outperforms other pretraining techniques *and* a randomly initialized model with a 16% improvement in PR AUC. Notably, the

model pretrained on the “Same User” task does significantly worse than the others. One plausible explanation is that by learning to embed windows from the same user in the same region of the latent space, the model sacrifices its ability to distinguish “unusual” (e.g. flu positive) windows for a given user, since these windows would ordinarily be much further away in the latent space.

We additionally compared this pretraining-fine tuning approach to a multitask learning where both the pretraining and the target prediction objective are optimized *concurrently*. We repeated this comparison for each of the pretraining tasks (Section 4.6) in combination with the “Flu Symptoms” task. This strategy produced no meaningful improvement over the randomly initialized model, indicating that unsupervised pretraining is a superior paradigm for this setting. In addition, the pretraining-finetuning paradigm enables us to separate these two steps across two datasets, especially when the target dataset is relatively small. This is the focus of the next two experiments.

4.6.4 EXPERIMENT 3: **Transfer learning improves flu prediction performance on small datasets in a repeated simulation study**

Labeled behavioral data is often prohibitively expensive to collect, particularly in the context of public health where ground-truth labels require costly testing infrastructure and study management. Accordingly, many studies from prior work operate on data with on the order of dozen participants [Xu et al., 2021]. In this regard, one promising application of generalizable self-supervised pretraining is that models could leverage large unlabeled datasets to improve predictive power in settings with limited labeled training data. Here, we repeatedly simulate such settings to robustly investigate whether such transfer learning leads to performance improvements.

First, we isolate all 4,989 study participants who never tested positive for the flu. We treat this set as a large, unlabeled dataset which we use to pretrain our model on the self-

	XGBoost	Our Model
Zero-shot PR AUC	0.005	0.018
Zero-shot ROC AUC	0.51	0.68

Table 4.4: Performance on zero-shot COVID-19 Prediction (Section 4.6.5). Our model’s superior performance shows that CNN-Transformers pretrained on the “Domain Inspired Features” task (Section 4.6) learn generalizable features.

supervised “Daily Features” task (Section 4.6). We then take the remaining 206 users who *did* test positive at some point during the study, and randomly split this set into twenty folds of ten or eleven users each. This ensures that source and target domain share no participants in common. We provide an overview of this split in Figure 4.6(a).

Next, for each fold we finetune the model on the supervised “Flu Positivity” task using data from the fold, and evaluate it on the users in the remaining nineteen folds. We choose this task as it mirrors the zero shot setting in the external dataset of EXPERIMENT 4. In both of these settings, all test subjects tested positive at some point and the predictive model attempts to predict on which day they do so. This process simulates finetuning the model with fewer than a dozen users’ data. We compare this approach to two non-pretrained models that only have access to the smaller target domain dataset: CNN-Transformer, and XGBoost trained on manually defined features (Table 4.3).

Results. We find that our pretrained model outperforms non-pretrained models on the “Flu Positivity” task when trained on fewer than a dozen participants (Figure 4.6(b)). Pre-training alone increases average performance from 0.626 ROC AUC to 0.665, and 0.017 PR AUC to 0.021 (both $p < 0.05$, Mann-Whitney U). This indicates that Our Model can learn generalizable features from unlabeled data.

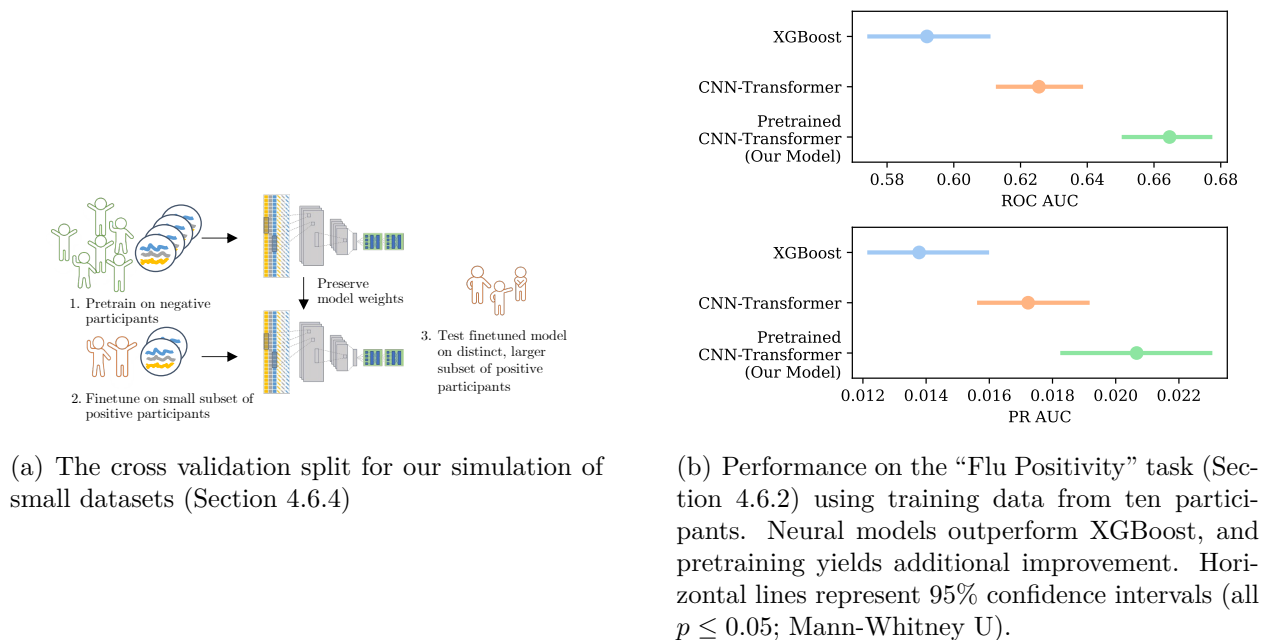


Figure 4.6: Simulating “small data” scenarios using training data from only ten participants (Section 4.6.4).

4.6.5 EXPERIMENT 4: Zero-shot COVID-19 prediction in a small external dataset

It is plausible that a self-supervised pretrained model, which in Experiment 1-3 showed good performance on flu related tasks, could support non-random predictive performance in a zero shot setting for COVID-19? Both diseases are respiratory viral infections and may trigger similar behavioral and physiological responses (e.g., a change in resting heart rate around symptom onset [Shapiro et al., 2021]). We use a small, independently collected dataset of Fitbit recordings and COVID-19 test results to show that our model can learn representations which generalize to entirely unseen diseases. This dataset contains 1470 total days of data for 32 individuals who tested positive with COVID-19 [Mishra et al., 2020]. The original study uses a retrospective prediction task with no train/test split, and so it is not possible to make a direct comparison between our model and theirs, but this dataset allows

us to test performance on an unseen disease.

We pretrain our model with the “Domain Inspired Features” task (Section 4.6) and finetune it on the “Predict Flu Positivity” task (Section 4.6). Note that this is the same configuration as “our model” in Table 4.2. Then, with no additional supervision we use the model to predict COVID-19 positivity in the small, external dataset. As a zero-shot baseline we calculate a set of day-level features (Table 4.5) from these data “and” our original flu dataset (Section 4.2) and train XGBoost with these features on the “Flu Prediction” task. Neither Our Model nor the XGBoost baseline is exposed to any data from the COVID-19 dataset during training.

Results. our model outperforms XGBoost on this zero-shot task, achieving 0.68 ROC AUC, while XGBoost predicts at 0.51 ROC AUC (random chance) (Table 4.4). This illustrates the feasibility of pretrained CNN-Transformers for novel disease prediction.

Feature	Description
Resting HR 95 th Pct	95 th percentile of resting HR
Resting HR 50 th Pct	50 th percentile of resting HR
Resting HR std.	Standard deviation of resting HR
Awake HR 95 th Pct	95 th percentile of HR while awake
Steps Streak 95 th Pct	95 th percentile of continuous steps
Steps Streak 50 th Pct	50 th percentile of continuous steps
Total Minutes in Bed	Number of minutes spent in bed
Sleep Minutes	Number of minutes spent asleep
Total Steps	Total number of steps
Missing HR	Indicator for missing HR data
Missing Sleep	Indicator for missing sleep data
Missing Steps	Indicator for missing steps
Missing Day	Indicator for missing all data

Table 4.5: Summary of manually defined features used for XGBoost baseline in the zero shot experiment (Section 4.6.5) calculated for every user and on each day. “Missing” features are binary variables which are 1 if more than one hour of data is missing, and 0 otherwise.

4.7 Conclusion

During the COVID-19 pandemic over-the-counter antigen tests have been effective in surveillance testing, but the frequency of tests, more so than their sensitivity, remains a barrier

to success in mitigating spread [Larremore et al., 2020]. This chapter presents a framework for evaluating mobile sensing methods for frequently predicting respiratory viral infections. While there are limitations to this study, our results show performance on par with COVID-19 rapid diagnostic tests in similar surveillance settings. More research is needed to demonstrate similar performance levels in larger studies. Nonetheless, our findings suggest that mobile sensing predictions can complement rapid antigen testing or trigger additional testing. Our results indicate that pretraining, transformer self-attention, modeling missing data, and transfer learning are effective techniques in learning generalizable behavioral representations for mobile sensing.

Chapter 5

MULTIVERSE: MINING COLLECTIVE DATA SCIENCE KNOWLEDGE FROM CODE ON THE WEB TO SUGGEST ALTERNATIVE ANALYSIS APPROACHES

This work was previously published at KDD '21.

A recent study asked twenty-nine teams of well-trained data analysts a narrowly phrased question: do referees give penalties to dark-skinned soccer players at a higher rate than light-skinned players [Silberzahn et al., 2018]? Surprisingly, while individual teams arrived at highly confident conclusions, there was no overall consensus among these teams.

Crucially, despite access to identical datasets and no appreciable technical errors, teams arrived at their conclusions through unique analyses which varied in their independent and dependent variables, model selection, statistical assumptions, and more.

Multiverse Analysis is an emerging concept in statistics and meta-science which attempts to describe this phenomenon by demonstrating that analysts must navigate a series of “decision points” in order to draw conclusions from data [Gelman and Loken, 2014, Steegen et al., 2016]. While each of the choices made at a decision point (such as setting a threshold or specifying a model) may be entirely reasonable and defensible, so may many of its alternatives. The resulting set of options produces a “garden of forking paths” from which analysts traditionally select only a single path from raw data to results [Gelman and Loken, 2014].

A growing body of work has identified the large multiverse of possible analyses as a significant contributor to the reproducibility crisis, since even minor changes to the analytical path can

decision point, formulating code snippets which provide an alternate analysis) (Chapter 5, Section 5.2). While this formalization permits us to programmatically support Multiverse Analysis, as of yet no large dataset of Multiverse Analyses exists with which to train such a model.

Here we turn to the collective expertise of analysts on the web in the form of popular data science competitions, where thousands of analysts work on the same task. A central idea of this paper is that small differences between submissions to each of these competitions contain potential analysis alternatives, allowing us to create a corpus of 70k alternatives from 40k submissions without additional costly expert supervision (subsection 5.3.1). For example, an analyst might change model definitions or hyperparameters between submissions to explore their impact on the analysis outcome.

A second idea is that code libraries tend to be intentionally designed to represent semantic relationships between objects and functions commonly used in data analysis. For example, `sklearn.cluster.KMeans` and `sklearn.cluster.DBSCAN` are defined in the same submodule (`clustering`) and could be reasonable alternatives to one another. We build on this idea by extracting a large graph (called a “library graph”) of these relationships to learn suitable code representations (subsection 5.3.2).

Leveraging this dataset representing collective data analysis expertise, we propose a novel neural architecture that is able to (1) detect decision points in analysis code (subsubsection 5.4.3), and (2) generate potential alternatives for the analyst to consider (subsubsection 5.4.3). We additionally (3) integrate structural library information using a graph neural network approach that informs the learned code representations (subsubsection 5.4.3). Further, we propose a formulation of beam search, called “Span-Aware Beam Search”, that limits the generation of new code to decision points, and leaves surrounding code unchanged, improving performance in *Alternative Generation*. Figure 5.1 shows examples of MULTIVERSE’s predictions, which correctly suggest alternative models, parameters, and thresholds.

We evaluate MULTIVERSE’s performance on the tasks defined in Chapter 5, Section 5.2. We show that MULTIVERSE achieves up to 0.81 ROC AUC on locating decision points in the “Decision Point Classification” task, and 88.7% ROUGE-L-F1 on the “Alternative Generation” task (Chapter 5, Section 5.5). We conduct an ablation study to show that MULTIVERSE’s library graph , Span-Aware Beam Search, and multitask formulation all strictly improve performance on Decision Point Classification ROC AUC and Alternative Generation ROUGE-L. We also evaluate our model against comparable seq2seq models for code, showing that it performs two to three times better than those models on Alternative Generation (e.g. from 29.2% ROUGE-L Precision to 93.3%). To simulate the setting where an analyst has already identified their decision points, we pass information about the location of decision points to MULTIVERSE and show that under this condition performance improves from 50.3% to 73.4% GLEU. Finally, we show through a human evaluation that real data analysts find MULTIVERSE’s alternatives to be more reasonable, acceptable, and syntactically correct than alternatives from comparable baselines, including other transformer-based seq2seq models.

Our work shows the feasibility of learning to recommend alternative analyses by mining collective data science knowledge from the web and has implications for improving reproducibility by supporting Multiverse Analysis.

5.1 Related Work

Reproducibility & The Data Analysis Multiverse is similar but distinct from meta-learning in that it attempts to describe *all* reasonable alternatives, instead of trying to discover the single “best” one. Prior work on the Data Analysis Multiverse has shown that analysts’ decisions can limit reproducibility in individual studies [Gelman and Loken, 2014], are frequently limited by experience and social pressure (e.g. pressure from a field to perform a “standard” analysis plan rather than another) [Liu et al., 2020b], and can produce

drastically different results on the same task [Silberzahn et al., 2018]. Researchers have developed tools to help analysts visualize alternative pathways [Liu et al., 2020a, Simonsohn et al., 2015], log versions of their own work [Kery et al., 2017, 2019], and detect false positives [LeBel et al., 2018]. Prior interviews have indicated that even seasoned data analysts struggle to develop multiverse alternatives [Liu et al., 2020b]. Therefore, we build on this work and propose a potential solution that supports analysts by developing the first model which can automatically suggest multiverse decision points and alternatives.

Seq2seq for Code. Sequence to sequence (seq2seq) models take a sequence (such as natural language or code) as input, and generate a corresponding sequence as output. These models have achieved significant results on complex tasks like machine translation [Bahdanau et al., 2014, Weng et al., 2019], speech recognition [Povey and et. al, 2011] and search [Hughes et al., 2019]. One area focuses on learning to represent edits, often involving common fixes for grammar, clarity and style [Fu et al., 2017, Li et al., 2017, Chollampatt and Ng, 2018].

Deep learning has recently become a powerful tool to apply seq2seq tasks to code. Typical seq2seq tasks for code include bug fixes [Chen et al., 2019], code transformation [Tufano et al., 2019], bug localization [Lam et al., 2015], API usage generation [Gu et al., 2016], etc. SequenceR [Chen et al., 2019] uses an LSTM encoder-decoder model with attention and copy mechanisms to generate simple one line fixes for bugs. Neural Code Translator [Tufano et al., 2019] uses an encoder-decoder recurrent neural network (RNN) to learn code changes before and after pull requests on Github. Lam et al. [Lam et al., 2015] use a revised Vector Space Model for bug localization, creating a representation to relate terms in a bug report to source code tokens. In contrast to these methods that separate the tasks of token classification and code generation, we propose a joint learning method for both Decision Point Classification and Alternative Generation and demonstrate that this multitask framework leads to improved performance (Chapter 5, Section 5.5). Furthermore, by using a BPE Tokenizer and a transformer architecture, we model an open vocabulary and support longer sequences than these alternatives.

Graph Representation Learning. Graph representation learning is a field that aims to embed nodes, edges [Grover and Leskovec, 2016], sub-graphs [et. al, 2018], and full graphs [Ying et al., 2018, Mohammadshahi and Henderson, 2020] in a high dimensional vector space that captures desirable properties of the original graph, such as node distance, hyperbolicity, or local neighborhoods. In this paper, we incorporate information from external code libraries by jointly learning representations of library structure and tokens from code in order to improve performance (subsubsection 5.4.3). Of particular interest to this paper is distortion, which describes the degree to which the pairwise distance between node embeddings reflects their distance in the original graph [Chami et al., 2019, 2020].

5.2 Multiverse Analysis Tasks

When a data analyst explores alternative formulations of their analysis they must identify likely decision points in their code and then construct a set of reasonable alternatives for each decision point. These distinct functions could be performed jointly (as in a multitask framework), or sequentially (first identifying decision points and only then providing alternatives). Furthermore, a user may have some strong prior belief about the location of decision points, or they could require substantial guidance to detect them (*e.g.*, due to a lack of experience with alternative approaches).

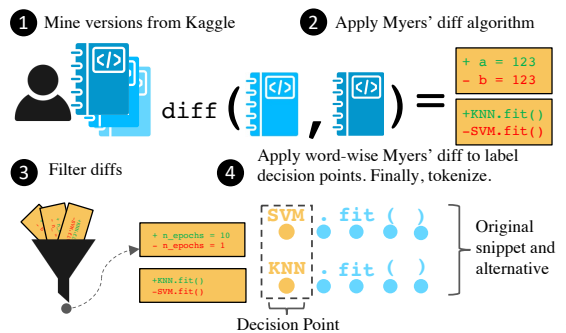


Figure 5.2: Our process for mining decision points from Kaggle notebooks.

In this paper we propose to formulate the goal of providing alternative analyses as two distinct prediction tasks: *Decision Point Classification* and *Alternative Generation*. In doing so, we are able to explore not only our model’s ability to provide useful alternatives, but

also unlock two other avenues for analysis. First, in Chapter 5, Section 5.5 we show that models that solve these tasks independently or treat our goal as a pure seq2seq task perform worse than a multitask model which jointly takes both objectives into account. Second, our multitask framework allows us to bound the model’s performance under the condition that it has additional information about the location of the decision points (subsection 5.5.3). This experimental setting mirrors the common paradigm where an analyst has some a priori intuition about which parts of their code they want to “multiverse”, that is, for which parts they would like to see alternatives.

5.2.1 Decision Point Classification

Creating an alternative analysis frequently entails making small edits to existing code, such as changing a hyper-parameter or substituting one statistical model for another. This identification process is often non-trivial, even for human analysts, as knowing which edits to code are likely to materially alter results requires a deep understanding of the underlying methods [Liu et al., 2020b]. We are therefore interested in classifying spans of code which are likely to belong to decision points, and whose modification therefore alters the final result. More formally, given a token vocabulary V , an input code sequence $x = \{x_t \in V\}, t = 0..N$, and a code alternative $y = \{y_t \in V\}, t = 0..N$, we would like to classify the set of sub-sequences in x that do not appear in y : x^{DP} (these are the subsequences that changed and may constitute a decision point and corresponding alternative). We also define y^{DP} as the analogous set of subsequences that do not appear in x .

5.2.2 Alternative Generation

When a data analyst creates an alternative analysis, they must not only locate decision points, but also write code to formulate their alternative approach. Like decision point classification, alternative generation is difficult even for seasoned analysts because it requires a broad knowledge of related statistical methods and creativity to imagine different formu-

lations of the same strategy. We model this process as a sequence to sequence (seq2seq) prediction task, whereby an alternative code snippet can be thought of as a machine translation of a corresponding snippet from the original analysis. More formally, we train our model to produce a code alternative $y = \{y_t \in V\}, t = 0..N$. Our goal in this machine translation task is find some \hat{y} that maximizes $P(\hat{y}|x) = \prod_{t=0}^{t=N} P(\hat{y}_t|x_0...x_N, \hat{y}_0... \hat{y}_{t-1})$.

5.3 Key Ideas

Next we describe two key ideas of this paper. First, we motivate and support our hypothesis that a carefully filtered set of Kaggle submissions represent alternative analyses. Second, we outline how the graphical structure of code libraries can be mined and incorporated into a model to help it learn relationships between relevant semantic entities in code.

5.3.1 Some Small Differences between Kaggle Submissions Constitute Alternative Analysis Approaches

On the popular data science competition website kaggle.com users compete against each other to answer questions with data, often for cash prizes. While Kaggle is best known for its machine learning competitions, users also clean, visualize, explore, and test data through hosted Jupyter Notebooks. Critically, Kaggle employs a version control system to publicly host all prior versions of public scripts. A key insight of this paper is that when a user submits multiple versions of their analysis using this feature, non-trivial edits between submissions can be considered as attempts to “multiverse” their approach. We do not claim that these edits represent “better” or more “correct” analyses, rather we argue that they represent code snippets with a common goal set by a user’s intention. As a user updates their submissions with new data cleaning methods, models, and evaluation metrics they are in effect expressing an alternate formulation of their own analysis. These updates are very common as analysts compete with each other to complete these tasks. Furthermore, since the site contains multiple users’ submissions to the same problems, these submissions may share a common

context. Taken as a whole, these scripts represent the efforts of tens of thousands of users to solve our tasks of Decision Point Classification and Alternative Generation (Chapter 5, Section 5.2). We detail our process in for mining alternatives in Figure 5.2.

Method. We crawled all versions of all public submissions to all competitions on Kaggle, yielding 48k submissions with an average of 9.3 versions of each script, or 450k scripts in total. Since Jupyter has more than 8 million users and is the most popular IDE among data analysts, we focus on Python notebook cells as our unit of analysis [JetBrains, 2018, Kelley and Granger, 2017]. However, the entirety the method presented in this paper could be applied to any unit of analysis (e.g. function declarations, individual lines), and any other language that allows imports from external libraries (e.g. Julia, R, Go).

Processing and Data Filtering. We then used the Myers’ diff algorithm (which is commonly used to compute git merges) to find edits between sequential versions of submissions [Myers, 1986]. In order to help our model interpret context, we include a line of unchanged code above and below each diff. A manual inspection of the dataset showed that a portion of these diffs were trivial edits such as changing a model’s checkpoint directory or reformatting code for readability. To focus our method on the types of edits that most reasonably constitute a multiverse analysis, we filtered out diffs whose edits were changes to whitespace, changes to I/O operations (e.g. `pd.read_csv("v1.csv")` \rightarrow `pd.read_json("v2.json")`), diffs that simply rearranged code without changing its semantics (e.g. `a + b` \rightarrow `b + a`), diffs that appeared to handle plotting (e.g. `fig.set_size(10,5)` \rightarrow `fig.set_size(10,10)`), and diffs that simply renamed variables (e.g. `clf = KNN()` \rightarrow `model = KNN()`). Initial explorations showed that most decision points and alternatives lead to changes of individual functions, function arguments, and typically involve very few lines of code. Therefore, we remove all diffs whose total size (including context) is more than five lines, leaving us with a final dataset of 70k pairs of original snippets and their alternatives from 40k submissions. For the Decision Point Classification task (subsection 5.2.1), we again apply Myers’ algorithm at the word level between the input x and its alternative y to find its x^{DP} and y^{DP} .

Decision Point	Original	Alternative
Missing Data	<code>train.csv.fillna(method = 'ffill')</code>	<code>train.csv.dropna().reset_index()</code>
Outliers	<code>max = np.quantile(duration.values(),0.9)</code>	<code>max = np.quantile(duration.values(),0.8)</code>
Filtering	<code>weapon = train[train["weaponsAcquired"]<25]</code>	<code>weapon = train[train["weaponsAcquired"]<=20]</code>
Data Type	<code>cat=KBinsDiscretize(encode='original')</code>	<code>cat=KBinsDiscretizer(encode = 'ordinal')</code>
Assumptions	<code>pvals = ttest_ind(sales)</code>	<code>pvals = ttest_ind(sales,equal_vars=True)</code>
Variable Selection	<code>smf.ols('SALARY.MILLIONS ~ WINS.RPM')</code>	<code>smf.ols('SALARY.MILLIONS ~ POINTS')</code>
Model Selection	<code>SVClr = SVC()</code>	<code>KNNclr = KNeighborsClassifier()</code>
Estimation Method	<code>make_pipeline(StandardScaler(), RidgeCV)</code>	<code>make_pipeline(RobustScaler(), RidgeCV)</code>
Inference Criteria	<code>final_vars = results.pvalues <=0.25</code>	<code>final_vars = results.pvalues <=0.15</code>

Table 5.1: Taxonomy of types of decision points in our dataset, with real examples from Kaggle of each type of decision point.

Validation. Data analyses include many different types of decisions. While we expect that decision type classifications will continue to evolve, possibly beyond the scope of our dataset, Wicherts et. al [Wicherts et al., 2016] contribute a useful taxonomy of nine different decision point types. We find that our dataset includes examples for all nine of these decision point types. Table 5.1 in shows this taxonomy of decision points with accompanying examples from our dataset, which demonstrates that our method of mining examples from Kaggle covers a diversity of decision points and alternatives. Furthermore, a human evaluation demonstrates that experienced data analysts are likely to accept machine-generated alternatives derived from this corpus (subsection 5.5.4).

5.3.2 Code Libraries Represent Semantic Relationships

When developers build code libraries, they often organize their projects so that semantically related function and class definitions are contained within the same package or module. For example, in the popular Python scientific computing library `scipy`, the t-test (`scipy.stats.ttest_ind`) is defined in the same module as Mann-Whitney U (`scipy.stats.mannwhitneyu`), one of its non-parametric alternatives. To train our model to learn this information, we capture a graph of these relationships by mining libraries for their structure. We note that while we describe a method for mining this structure from Python, a similar process can be followed for any other object-oriented language like R or Julia.

Method. First, we mine import statements from our dataset (subsection 5.3.1) to find the

top ten most frequently used libraries. For each library, we start with the file invoked by the highest level import statement (e.g. `import scipy`) and then recursively create a node in our graph for each file, and then neighbor nodes for each sub-module it imports and each function or class defined in the file. For each class definition, we create neighbor nodes for each of its methods. For example, the class `scipy.stats` is the neighbor of `scipy`, and has a set of neighbors including `scipy.stats.mannwhitneyu` and `scipy.stats.ttest`. We note that this method is not guaranteed to create a tree, since some sub-modules might be imported from multiple files. Since functions are rarely referenced by their full import path, at each node u we store $name_u$, the name of the class, function, or method. For example, the name of `scipy.stats.mannwhitneyu` is `mannwhitneyu`. Later, we will use these names to relate nodes of the library graph to tokens in code snippets (subsection 5.4.3).

Processing and Data Filtering. Many nodes in the unprocessed graph (e.g. `numpy.testing.decorate_methods`) are unlikely to be of interest to analysts, and so we remove nodes if they have a parent called `testing` or `test` and all private methods. This results in a graph with over 60k nodes representing submodules, functions, and classes in popular data analysis libraries.

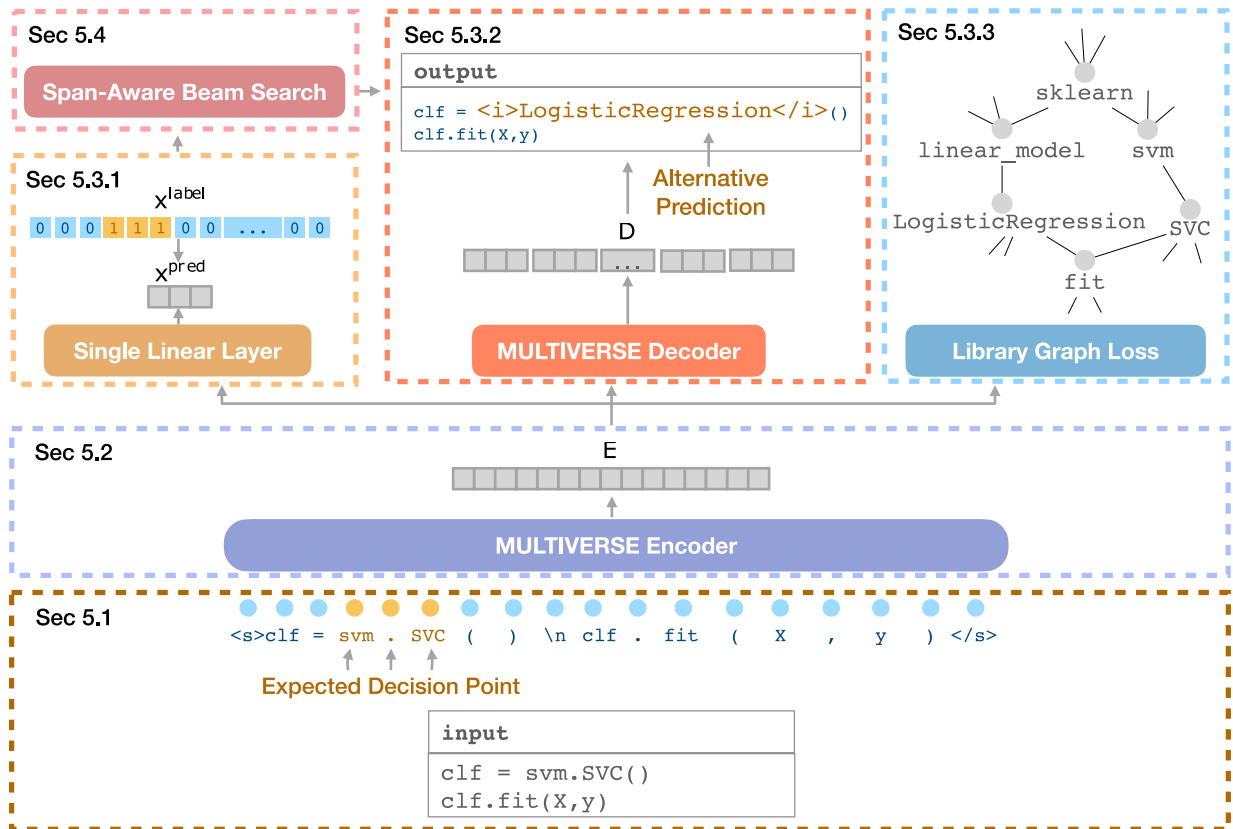


Figure 5.3: Our MULTIVERSE model, which combines Decision Point Classification, Alternative Generation and library embedding into a multitask objective.

5.4 Methods

Here, we describe the core components of MULTIVERSE (Figure 5.3). We model code as a sequence of tokens, allowing for increased flexibility with respect to partial or syntactically incorrect snippets (subsection 5.4.1). Our model combines a bidirectional transformer encoder with a left to right decoder to incorporate context from the whole input sequence (subsection 5.4.2). Furthermore, we present a series of objectives, which we ultimately combine into one multitask objective (subsection 5.4.3). Finally, we motivate and develop a novel seq2seq decoding strategy that can decide to simply copy input sequences into the output (subsection 5.4.4).

Notation. Formally, given a token vocabulary V , and an input code sequence $x = \{x_t \in V\}, t = 0..N$, we train our model to produce a code alternative $y = \{y_t \in V\}, t = 0..N$. Our goal in conditional generation is to estimate:

$$P(y|x) = \prod_{t=0}^{t=N} P(y_t|x_0...x_N, y_0...y_{t-1}) \quad (5.1)$$

5.4.1 Model Inputs

MULTIVERSE represents code as a series of tokens, allowing it to handle messy, potentially incomplete code that might otherwise be unparseable to an AST-based method. Furthermore, by using a byte-wise pair encoding (BPE) tokenizer (which splits rare and unknown words into more commonly occurring subtokens) we are able to model the large vocabularies that code corpora are known for [Svyatkovskiy et al., 2020, Sennrich et al., 2015]. We also insert special tokens `<INSERTED>` and `</INSERTED>` at the beginning and end of subsequences in y^{DP} (subsection 5.2.1), which we use in Span-Aware Beam Search (subsection 5.4.4). Crucially, we only use these special tokens in training labels, and *not* in inputs, to simulate our use case where the user has no knowledge about the location of decision points in their code.

5.4.2 Bidirectional Encoder and Left-To-Right Decoder

MULTIVERSE combines a bidirectional transformer encoder with a left-to-right autoregressive decoder, similar to BART [Lewis et al., 2019b]. In MULTIVERSE and associated ablation studies (subsection 5.5.2) we use four layers in both the encoder and the decoder, and four attention heads in each.

Encoder. MULTIVERSE uses a bidirectional transformer encoder with multi-head attention to calculate representations of the tokens in the input code sequence:

$$h_0 = W_e \cdot x + W_p \tag{5.2}$$

$$h_i = \text{TransformerBlock}(h_{i-1}), i \in 1 \dots n \tag{5.3}$$

$$h_i^{norm} = \text{LayerNorm}(h_i) \tag{5.4}$$

$$E = h_n^{norm} \tag{5.5}$$

Where $W_e \in \mathbb{R}^{|V| \times d_e}$ is an initial embedding matrix and $W_p \in \mathbb{R}^{N \times d_e}$ is a learned positional embedding matrix. We experimented with embedding sizes, and found $d_e = 128$ to perform as well as larger embedding dimensions. The final input sequence embedding E is passed forward to the decoder.

Decoder. The left-to-right auto-regressive decoder generates a distribution over possible new tokens given the input context x and previously decoded tokens. During training, the decoder takes the alternative y shifted to the left (y^{shift}) as input, such that $y_i^{shift} = y_{i-1} \forall i \in 1..N, y_0^{shift} = \langle \text{START_OF_SEQUENCE} \rangle$. The decoder incorporates information from the input

context by attending to the encoder output embedding E in each layer.

$$g_0 = W_e \cdot y^{shift} + W_p \quad (5.6)$$

$$g_i = \text{TransformerBlock}(g_{i-1}, E), i \in 1 \dots n \quad (5.7)$$

$$g_i^{norm} = \text{LayerNorm}(g_i) \quad (5.8)$$

$$D = g_n^{norm} \quad (5.9)$$

Finally, we calculate:

$$P(y_t | x_0 \dots x_N, y_0 \dots y_{t-1}) = \text{softmax}(D_t) \quad (5.10)$$

5.4.3 Objectives

Decision Point Classification Loss

To address our decision point classification task (subsection 5.2.1) we use a token-level classification head composed of a single linear layer with cross entropy loss over the encoder's final hidden states E :

$$x_t^{pred} = \text{SoftMax}(\text{LinearLayer}(E_t)) \quad (5.11)$$

$$x_t^{label} = \begin{cases} 1 & \text{if } t \in x^{DP} \\ 0 & \text{otherwise} \end{cases} \quad (5.12)$$

$$\mathcal{L}_{cls}(x^{pred}, x^{label}) = \text{CrossEntropy}(x^{pred}, x^{label}, \gamma) \quad (5.13)$$

Where γ is a positive class weight we adopt to combat class imbalance, and E_t is the output embedding of the t^{th} input token. In practice, since decision point tokens represent approximately one tenth of the total tokens in our corpus, we set $\gamma = 10$.

Alternative Generation Loss

To train our model to generate alternative analyses (subsection 5.2.2) we adopt a standard seq2seq cross entropy objective over the output of the model’s decoder:

$$\mathcal{L}_{alt}(D, y) = \text{CrossEntropy}(\text{SoftMax}(D), y) \quad (5.14)$$

Library Graph Loss

In order to teach our model about the semantic relationships in libraries, we jointly embed our library graph with tokens from the input sequence. In particular, we optimize for graph distortion, which captures how the pairwise graph distances between n nodes in a graph U differ from their distance under a metric d_V in an embedding space V , where $g : U \rightarrow V$ is a function that maps nodes to their embeddings:

$$Dist(g) = \frac{1}{\binom{n}{2}} \left(\sum_{u,v \in U; u \neq v} \frac{|d_V(g(u), g(v)) - d_U(u, v)|}{d_U(u, v)} \right) \quad (5.15)$$

The gradient of $Dist$ is undefined at $d_V(g(u), g(v)) = d_U(u, v)$, and numerically unstable elsewhere, and so prior work has focused on finding embeddings that minimize related objectives. In our setting, we define $d_U(x_t, x_j)$ as the undirected shortest path through the library graph between any two nodes u, v whose names $name_u, name_v$ can be tokenized to include x_t and x_j , respectively. For example, `sklearn`’s KNN and SVM modules both have methods named `predict_proba`, and so tokens corresponding to KNN and SVM would have a library graph distance of two. On average, 45% of the tokens in an input sequence can be related to the graph through this method. We define library embedding loss as

$$\mathcal{L}_{lib} = \beta \frac{1}{\binom{|x^g|}{2}} \left(\sum_{t,j \in x^g; t \neq j} \left| \frac{d_V(E_t, E_j)^2}{d_U(x_t, x_j)^2} - 1 \right| \right) \quad (5.16)$$

where x^g is the set of nodes in the graph whose corresponding tokens appear in the input, $E_u \in \mathbb{R}^{d_e}$ is the encoder embedding of token u : E_u , d_V is a metric over the embedding space, and β is a scaling factor. Since large graphs with low hyperbolicity like ours have been shown to exhibit the lowest distortion when embedded in euclidean space, we use the L^2 norm as d_V [Chami et al., 2019]. Intuitively, this objective is minimized by embedding input tokens in a space where they are close to their neighbors in the library graph.

Multitask Objective. In MULTIVERSE we optimize each of these objectives jointly by adding them into a single multitask objective:

$$\mathcal{L} = \lambda_{cls}\mathcal{L}_{cls} + \lambda_{alt}\mathcal{L}_{alt} + \lambda_{lib}\mathcal{L}_{lib} \quad (5.17)$$

where each λ represents a constant weight on each term.

5.4.4 Span-Aware Beam Search

Since decision points typically span only a fraction of the total tokens in the input sequence, most input tokens (90% in our dataset) also appear in the output. For example, in the code sequence `clf = LogisticRegression(X,y)`, only the tokens corresponding to `LogisticRegression` are likely to belong to a decision point (and are therefore likely to change in the alternative), while most reasonable alternatives would include the tokens corresponding to `clf =` and `(X,y)`. We hypothesize that a model could perform better if it only had to generate in the “holes” between non-decision points.

To test this theory, we propose a modified version of beam search that forces the model to preserve tokens from the input if the decision point classifier indicates that they are unlikely to change (subsection 5.4.3). Concretely, for each token if the softmax output of the positive class in the the decision point classification head (Equation 5.11) is above some threshold, then we only generate new tokens between this decision point and the next. Then, the beams proceed with standard generation until they hit a `</INSERTED>` token (at which point the

beams are again constricted) or an `<END_OF_SEQUENCE>` token (generation ends).

5.5 Evaluation

MULTIVERSE achieves a ROC AUC of 0.814 on the decision point classification task and a ROUGE-L F1 of 93.3% on the alternative generation task (Chapter 5, Section 5.2). We additionally compare MULTIVERSE’s performance on these tasks to ablations of the model and several state-of-the-art neural and non-neural baselines. Furthermore, to directly evaluate how useful MULTIVERSE’s alternatives are to end users, we conduct a human evaluation (subsection 5.5.4) and show that MULTIVERSE performs better than various baselines in syntactical correctness, reasonableness, and rate of end-user acceptance.

In the following evaluations MULTIVERSE is trained on submissions to 95% of competitions and is evaluated on the remaining 5% such that there is no overlap in submissions or competitions between the train and test sets. This ensures that the model never saw test-time analysis tasks during training, creating a conservative but realistic evaluation paradigm.

5.5.1 Task 1: Decision Point Classification

Baselines. To evaluate our model’s performance on the decision point classification task relative to simple non-neural baselines and models from prior work, we benchmark MULTI-

Model	Accuracy	F1	ROC AUC
MULTIVERSE	64.4%	32.5%	0.814
MULTIVERSE No Decoder	88.7%	45.7%	0.831
Sample from Train Set	15.5%	17.4%	0.517
Random Guessing	83.1%	9.1%	0.500
BiLSTM Huang et al. [2015]	91.2%	38.3%	0.828
BiLSTM-CRF Huang et al. [2015]	89.8%	40.7%	0.834

Table 5.2: Comparisons of MULTIVERSE with various neural models and non-neural baselines for Decision Point Classification.

Model	ROUGE-L			Span-ROUGE-L		
	Precision	Recall	F1	Precision	Recall	F1
MULTIVERSE	93.3%	85.9%	88.7%	30.5%	32.1%	25.6%
Neural Code Translator Tufano et al. [2019]	29.2%	12.3%	15.7%	4.7%	5.7%	4.0%
SequenceR Chen et al. [2019]	50.7%	24.9%	31.8%	5.3%	5.2%	4.1%
Copy Original Input	79.2%	76.7%	77.1%	0.0%	0.0%	0.0%
Randomly Replace Tokens	54.1%	55.0%	53.7%	1.3%	1.8%	1.3%
Tree Replace Tokens	15.7%	20.7%	17.3%	2.4%	8.4%	3.1%

Table 5.3: Comparisons of MULTIVERSE with various neural code seq2seq models and non-neural baselines. Our model consistently outperforms these baselines.

Model	Alternative Generation						Decision Point Classification			
	ROUGE-L			Span-ROUGE-L			GLEU	Accuracy	F1	ROC AUC
	Precision	Recall	F1	Precision	Recall	F1				
MULTIVERSE ¹	93.3%	85.9%	88.7%	30.5%	32.1%	25.6%	50.3%	64.7%	32.5%	0.814
No Library Graph ²	92.5%	79.1%	84.5%	29.5%	33.4%	22.5%	39.2%	64.8%	31.8%	0.800
No Span-Aware Beam Search ³	91.4%	83.2%	86.2%	33.7%	27.1%	26.1%	49.4%	65.3%	32.2%	0.787
No Multitask ⁴	87.3%	54.8%	64.7%	20.4%	19.6%	17.2%	11.8%	88.0%	44.0%	0.811
Alternative Generation Only ⁵	92.6%	84.5%	87.4%	31.7%	25.3%	24.1%	51.1%	N/A	N/A	N/A
MULTIVERSE- Full Information ⁶	95.8%	93.5%	94.2%	35.0%	30.9%	28.6%	73.4%	N/A	N/A	N/A

1. Our complete multitask model, which includes Span-Aware Beam Search and optimizes $\mathcal{L}_{cls} + \mathcal{L}_{alt} + \mathcal{L}_{lib}$
 2. A multitask model which includes Span-Aware Beam Search but not the library graph and optimizes $\mathcal{L}_{cls} + \mathcal{L}_{alt}$
 3. A multitask model which does not include Span-Aware Beam Search and optimizes $\mathcal{L}_{cls} + \mathcal{L}_{alt} + \mathcal{L}_{lib}$
 4. A model in which the MULTIVERSE encode and decode are trained independently
 5. A model which only optimizes \mathcal{L}_{alt}
 6. A model which receives perfect information about the location of decision points (§subsec:oracle).

Table 5.4: Comparison of various ablations of MULTIVERSE. We show that MULTIVERSE model performs best on the Alternative Generation task with respect to ROUGE-L, and that other ablations of MULTIVERSE perform best by a slim margin on Span-ROUGE-L metrics. Notably, MULTIVERSE outperforms all ablations when given access to information about the location of decision points (§subsec:oracle).

VERSE against the following. Full results are available in Table 5.2.

- **How well does MULTIVERSE’s encoder classify decision points on its own?**
To answer this question, we independently train the MULTIVERSE encoder without loss from the decoder. We note that this experiment is comparable to BERT [Devlin et al., 2019]. We denote this model as **MULTIVERSE No Decoder**.
- **How does random guessing perform on this decision point classification task?** To answer this question, we choose 10% (the proportion of decision points in the train set) of the token positions and take these tokens as decision points. We denote this model as **Random Guessing**.
- **How does a simple baseline that uses heuristics from the train set perform?**
To answer this question, we construct a baseline that classifies any token that was labeled as a decision point in the train set as a decision point in evaluation.
- **How well does a sequential neural network for sequence labeling in NLP work on our task?** To answer this question, we use a **BiLSTM** model as our baseline. LSTMs have been used to model source code in the past [Hellendoorn and

Devanbu, 2017]. This baseline is a multi-layer bi-directional LSTM (BiLSTM) where the input sequence is processed both forward and backward [Huang et al., 2015]. After concatenating the forward and backward hidden states for each LSTM layer, we pass them through a linear layer to make the predictions of which tokens are decision points.

- **BiLSTM-CRF** How well does a BiLSTM-CRF model for sequence labeling tasks work on our task? While BiLSTM only considers the likelihood of the word being a certain tag, a CRF also calculates the transition scores [Huang et al., 2015]. This baseline considers the likelihood of a token being a certain tag given the previous token.

Discussion. While our two BiLSTM baselines outperform MULTIVERSE on accuracy and by a slim margin on ROC AUC, MULTIVERSE No Decoder achieves the best F1. While these models perform better than MULTIVERSE, only MULTIVERSE can incorporate this information into down-stream Alternative Generation through its multitask objective and Span-Aware Beam Search. In the next section, we will see how this multitask objective enables significantly improved generations. Interestingly, this multitask objective seems to trade increased performance on Alternative Generation for decreased performance on Decision Point Classification. This is likely because MULTIVERSE learns to classify tokens to limit the chance that it makes an error on Alternative Generation.

5.5.2 Task 2: Alternative Generation

Model Ablations. In order to evaluate how our model’s individual components contribute to its overall performance, we train several ablations using different combinations of the components of its architecture. The results of the following studies are in Table 5.4.

- **MULTIVERSE** uses the bidirectional encoder and left-to-right decoder (subsection 5.4.2), the decision point classification loss \mathcal{L}_{cls} (subsubsection 5.4.3), the library graph loss \mathcal{L}_{lib} (subsubsection 5.4.3), and span-aware beam search (subsection 5.4.4). Formally, we optimize \mathcal{L} .

- **MULTIVERSE: No Library Graph** In this experiment, we try to understand how much of the model’s performance can be attributed to information mined from our graph of library structures (subsection 5.3.2). Accordingly, we optimize our model without the library embedding loss (subsubsection 5.4.3), but maintain the decision point classification loss (subsubsection 5.4.3) and the alternative generation loss (subsubsection 5.4.3). Formally, we optimize $\mathcal{L}_{cls} + \mathcal{L}_{lib}$.
- **MULTIVERSE: No Span-Aware Beam Search** How does our model perform without Span-Aware Beam Search, when it can not explicitly copy likely non-decision-point tokens from the input? Here, we generate with conventional beam search.
- **MULTIVERSE: No Multitask** How does our model perform if we split it by treating our multitask objective as two discrete tasks? We separately train the encoder on Decision Point Classification and the decoder on Alternative Generation, then use predictions from the encoder for Span-Aware Decoding.
- **MULTIVERSE: Alternative Generation Only** How well does our model perform if we decide not to include library embedding loss and decision point classification loss, and focus on Alternative Generation only? To answer this question, we remove the corresponding terms from our objective. This baseline is similar to seq2seq transformer models like BART [Lewis et al., 2019b]. Formally, we optimize \mathcal{L}_{alt} (subsubsection 5.4.3).

Baselines. We also compare MULTIVERSE to a set of baselines.

- **Neural Code Translator** How well does a seq2seq neural model designed for code perform on our task? To address this question, we compare MULTIVERSE against the Neural Code Translator model, which applies an RNN to learn code changes implemented by developers [Tufano et al., 2019].
- **SequenceR** How well does a bug fix neural model perform on our task? To answer this question, we compare MULTIVERSE against SequenceR, which uses a BiLSTM

encoder-decoder model to generate simple, small patches for buggy code [Chen et al., 2019].

- **Original Input** Since decision points and alternatives span only a small fraction of the total sequence, even a model repeating the original code sequences could achieve high ROUGE scores. Therefore, we construct a trivial baseline that completely copies the original input for reference.
- **Randomly Replace Tokens** How does a simple baseline that has no knowledge of where decision points are perform? To answer this question, we construct a trivial baseline that randomly replaces input tokens with other tokens in the vocabulary. Consistent with the proportion of decision points in the dataset, we choose 10% of the token positions at random. If the token is chosen, we replace it with a random token from the vocabulary.
- **Tree Replace Tokens** How well does a simple baseline perform that considers structural information of common libraries? To address this questions, we construct a baseline that considers library nodes as decision points and objects and functions that were defined in the same modules as alternatives. If the token appears in the library graph, we randomly replace it with a randomly sampled neighbor in the graph.

Metrics. We adopt the standard seq2seq machine translation metric ROUGE-L for evaluating our ablations and baselines on the alternative generation task [Lin, 2004]. In essence, ROUGE judges similarities between prediction and target sequences by measuring co-occurrences of common n-grams. Since original code snippets often share a significant portion of their tokens with their alternatives, (more than 90% in our corpus) ROUGE-L will inflate the performance of models that simply copy code from the input to the prediction without interpreting the significance of decision points. We therefore also introduce an additional metric “Span-ROUGE-L”, which is ROUGE-L calculated only over those tokens that belong to decision points: $\text{Span-ROUGE-L}(\hat{y}, y) = \text{ROUGE-L}(\hat{y}^{DP}, y^{DP})$.

We also include the GLEU metric, which was originally developed to evaluate machine-generated grammatical error corrections [Napoles et al., 2016]. Similar to ROUGE, GLEU measures n-gram overlap between a generated and ground truth sequence, but unlike ROUGE it penalizes models for predicting n-grams from the input sequence that should have changed in the output. To aggregate, we report the average of these metrics on each sequence in the dataset.

Discussion. MULTIVERSE performs up to three times better than other code-specific neural models, such as SequenceR [Chen et al., 2019] and Neural Code Translator [Bahdanau et al., 2014]. Ablations show that all components of our approach, library graph, Span-Aware Beam Search, and multitask objective, lead to significant performance improvements across most evaluation metrics. One exception is the Alternative Generation model, which performs marginally better than MULTIVERSE on GLEU. However, this model is incapable of providing predictions on the Decision Point Classification task. Importantly, as shown in the next section, providing these predictions through the multitask formulation significantly boosts performance for MULTIVERSE, from 50.3% to 73.4% GLEU.

5.5.3 Simulating Known Decision Points

What if an analyst knew the spans for which they wanted to generate alternatives? How much better could MULTIVERSE perform? To answer this question, we pass the ground truth decision point labels, x^{label} (subsection 5.4.1), to MULTIVERSE’s Span-Aware Beam Search (subsection 5.4.4). We find that under this condition MULTIVERSE’s GLEU Score increases significantly from 50.3% to 73.4%. Importantly, on Span-ROUGE F1 and Recall MULTIVERSE’s performance increases significantly from 25.6% to 28.5% and 30.5% to 35.0%. This suggests that if MULTIVERSE is given perfect information about the location of decision points by an analyst, it delivers better predictions within those decision points.

How much better does MULTIVERSE’s Decision Point classifier need to be to reap these

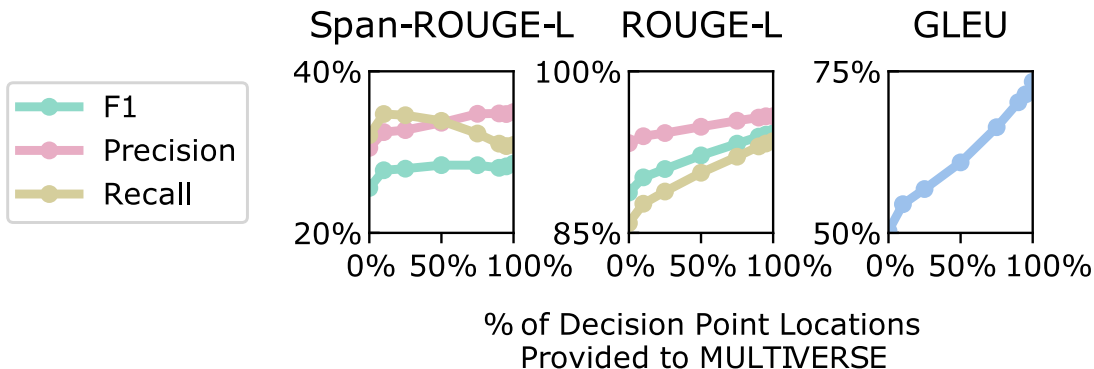


Figure 5.4: MULTIVERSE’s performance under increasing information about the location of decision points (subsection 5.5.3).

Method	Syntax	Reasonableness	Semantic Acceptance
MULTIVERSE(Our Method)	4.30	3.23	2.94
Alternative Generation Only	3.50	2.66	2.42
SequenceR [6]	2.17	1.48	1.42
Kendall’s W	0.664	0.534	0.462

Table 5.5: Mean scores from our user study (subsection 5.5.4). Five is the best possible score in each category. Our method significantly outperforms all baselines ($p < 0.01$, Wilcoxon’s Signed Rank). Analysts achieved moderate to substantial agreement on their ratings (Kendall’s W).

benefits? To answer this question, we take MULTIVERSE’s Decision Point Classification predictions and with a given probability change the token prediction to be equal to the ground truth x^{label} . We find that increases in Decision Point Classification accuracy are approximately linear with increases in performance on ROUGE, Span-ROUGE, and GLEU. Results are available in Figure 5.4. This shows that improvements in Decision Point Classification would benefit Alternative Generation as well, even without additional changes to Alternative Generation models.

5.5.4 Human Evaluation

Ultimately, we are most interested in how useful a model is to an analyst. In addition to the automatic evaluation, we perform a user study of MULTIVERSE’s predictions. Here we evaluate not just how well we do on automated metrics, but how well we perform on giving alternatives to analysts.

Five PhD-level data scientists with significant experience with Python for data science and machine learning were recruited as participants (co-authors were excluded from participation). The participants were asked to blindly evaluate outputs on their syntactical correctness, general “reasonableness” (defined as how well the result provided an alternative that a user might find useful) and semantic acceptance (defined as how likely each rater would be to accept the suggestion in the context of a multiverse analysis). The five participants evaluated 50 examples of alternates from MULTIVERSE and two state-of-the-art neural baselines including a standard seq2seq transformer-based model (comparable to BART [Lewis et al., 2019b]) and a neural code generation model [Tufano et al., 2019]. We chose these models for this evaluation because they had the best performance next to MULTIVERSE on Alternative Generation among the baselines (Table 5.3) and ablations (Table 5.4), respectively. Participants used a five point Likert scale for a total of $50 \times 3 \times 3 \times 5 = 2250$ ratings.

Notably, MULTIVERSE outperforms the baselines in all three criteria ($p < 0.01$, Wilcoxon’s Signed Rank Test). We observe moderate to substantial inter-rater reliability on all criteria (0.46-0.66, Kendall’s W). Results can be found in Table 5.5. This shows that MULTIVERSE’s predicted alternatives demonstrate higher syntactical correctness and reasonableness than comparable models. Furthermore, analysts are more accepting of MULTIVERSE’s suggestions.

5.6 Error Analysis

We conducted a qualitative error analysis of MULTIVERSE’s predictions to identify areas for future improvement. MULTIVERSE’s common errors can be grouped into three categories: degeneration, semantic failure, and input copying.

Degeneration. Degeneration is a common problem in seq2seq models where generation gets “stuck” in a loop. In one example alternative, MULTIVERSE predicts `y_train`, `y_test`, `y_train`, `y_test` until it reaches its maximum length threshold. One likely explanation is that `y_test` and `y_train` are the most likely tokens to follow one another under the language model. Solutions may include alternative sampling methods (such as top-k sampling or nucleus sampling) instead of beam search [Holtzman et al., 2020].

Semantic Failure. One feature of MULTIVERSE is that it can easily be trained on any computer language. However, this flexibility comes at the expense of built-in information about underlying language syntax. While MULTIVERSE scores highest against comparable models in syntactical correctness (subsection 5.5.4), we observe that the model still occasionally produces malformed Python code, such as `df =, pd.DataFrame()`. Prior work indicates that current-generation language models like ours can learn to generate nearly perfectly correct code given millions of training examples [Svyatkovskiy et al., 2020]. We believe that additional pre-training on large unsupervised corpora could mitigate this error.

Input Copying. In some cases, MULTIVERSE appears to have learned to exactly repeat the input sequence as its output. One explanation is that since input sequences in our dataset on average share 90% of their tokens with their corresponding output sequences, the model can score reasonably well by naively copying the input. Alternate formulations of cross entropy loss, such as frequency-aware cross entropy loss, have been shown to promote output diversity [Jiang et al., 2019], and could penalize MULTIVERSE for repeating its input during training. However, in an early version of the model we implemented this loss

function and saw no significant improvement in performance.

5.7 Conclusion

In this chapter we proposed two prediction tasks that support Multiverse Analysis [Gelman and Loken, 2014, Steegen et al., 2016], a novel practice aimed at improving reproducibility in data science. Identifying decision points and suggesting alternative analysis approaches were operationalized as a classification task and a sequence-to-sequence prediction task, respectively. We share datasets to support these tasks, based on mining decision points from kaggle and the graph structures of common libraries.

We showed that by formulating Multiverse Analysis as a multitask problem, our model and compares favorably to neural baselines from prior work on both the decision point classification and alternative generation tasks (Chapter 5, Section 4.3). Furthermore, if our model is given additional information about the location of decision points through Span-Aware Beam Search (guided by the analyst), its performance increases significantly beyond all other models, from 50.3% to 73.4% GLEU (subsection 5.5.3). Finally, we showed through a human evaluation that data analysts find our model’s alternatives to be more syntactically correct, reasonable, and acceptable than alternatives from other transformer and LSTM-based models.

Our work demonstrates the feasibility of learning to recommend alternative analyses by mining collective data science knowledge from the web and has implications for improving reproducibility by supporting multiverse analysis.

Chapter 6

CONCLUSION



Figure 6.1: An image of a dog (Franklin).

In this thesis, I presented three works that lay a foundation for multimodal models of time series and text. I precisely defined what it means for a model to reason about time series and demonstrated that current large language models perform no better than random chance on such tasks. I showed that, since some questions about time series can be answered by writing code, language model agents can assist in interpreting these data. Finally, I examined methods for pretraining encoders for time series data. Collectively, these contributions represent significant progress toward building automatic systems capable of natively understanding and analyzing time series data.

Currently, most time series interpretation relies on handcrafted analyses by domain experts. As the field progresses and multimodal models mature, I hope to see systems that can ingest and draw conclusions from large volumes of time series data. These systems could, for example, automatically detect and respond to patterns in market data, uncover hidden causal relationships in scientific experiments, or autonomously react to unforeseen changes in patients' health.

What barriers to this goal remain? One key obstacle is the absence of an “ImageNet moment” for time series. A comprehensive and powerful dataset of real-world time series data could provide the benchmark and training resources necessary for addressing the challenges described in Chapter 2. One promising research direction would involve carefully curating a naturalistic dataset of time series paired with text. However, this effort is fraught with challenges, as discussed in Chapter 6, Section 2.1. For instance, text and time series pairs are rare online, and many types of time series data (e.g., health or financial data) are sensitive in nature. Progress toward identifying or constructing such a dataset would be highly impactful. One potential solution could involve scaling the synthetic code-based generation scheme detailed in Chapter 6, Section 2.3.

Another significant challenge lies in embedding time series data in a manner compatible with existing language models. While my work in Chapter 4 offers some guidance on techniques for pretraining time series encoders, the problem remains far from resolved. Future research should focus on building a “CLIP [Radford et al., 2021] for time series”—a model that jointly learns to embed both time series and their corresponding textual descriptions. However, the feasibility of this approach is heavily dependent on resolving the data availability problem.

As shown in Chapter 4, some time series questions can be answered by writing code with LLM agents. While this method is effective in narrow domains, it is not clear that it represents the right approach for building general-purpose intelligent systems for time series.

To illustrate this limitation, consider how computer vision systems might function if they

relied solely on writing code to interpret images rather than using a learned encoder. How would an agent write a function to answer a simple question like, “Is there a dog in this image?” For that matter, how would a human? It is unlikely that this question could be answered through code alone, and yet the information is clearly present (Figure 6.1). Deep perception depends on the fundamental ability to natively process information, rather than relying on symbolic logic as an intermediate step.

Humans did not evolve to perceive time series signals in the same way we process visual information, but that does not mean machines should inherit this limitation. What hidden patterns might be lurking in time series data, invisible to human perception? Only time will tell.

BIBLIOGRAPHY

- Danielle Albers, Michael Correll, and Michael Gleicher. Task-driven evaluation of aggregation in time series visualization. In *SIGCHI*, 2014.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision, 2021.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. Technical Report arXiv:2010.11929, arXiv, June 2021. arXiv:2010.11929 [cs] type: article.

- Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General Perception with Iterative Attention. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4651–4664. PMLR, 2021.
- Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruva Ghosh, Jieyu Zhang, Eyal Orgad, Rahim Entezari, Giannis Daras, Sarah Pratt, Vivek Ramanujan, Yonatan Bitton, Kalyani Marathe, Stephen Mussmann, Richard Vencu, Mehdi Cherti, Ranjay Krishna, Pang Wei Koh, Olga Saukh, Alexander Ratner, Shuran Song, Hannaneh Hajishirzi, Ali Farhadi, Romain Beaumont, Sewoong Oh, Alex Dimakis, Jenia Jitsev, Yair Carmon, Vaishaal Shankar, and Ludwig Schmidt. DataComp: In search of the next generation of multimodal datasets, 2023.
- Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: A Visual Language Model for Few-Shot Learning, November 2022.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *NeurIPS*, 2023a.
- Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*, 2015.
- Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, J. Hockenmaier, and David Alexander Forsyth. Every picture tells a story: Generating sentences from images. In *European Conference on Computer Vision*, 2010.

Chang Wei Tan, Christoph Bergmeir, Francois Petitjean, and Geoffrey I Webb. Monash university, uea, ucr time series extrinsic regression archive. *arXiv preprint arXiv:2006.10996*, 2020.

Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

André Bauer, Marwin Züfle, Simon Eismann, Johannes Grohmann, Nikolas Herbst, and Samuel Kounev. Libra: A benchmark for time series forecasting methods. In *ICPE*, 2021.

Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, Eugene Byrne, Zachary Chavis, Joya Chen, Feng Cheng, Fu-Jen Chu, Sean Crane, Avijit Dasgupta, Jing Dong, María Escobar, Cristhian Forigua, Abrham Kahsay Gebreselasie, Sanjay Haresh, Jing Huang, Md Mohaiminul Islam, Suyog Dutt Jain, Rawal Khirodkar, Devansh Kukreja, Kevin J Liang, Jia-Wei Liu, Sagnik Majumder, Yongsen Mao, Miguel Martin, E. Mavroudi, Tushar Nagarajan, Francesco Ragusa, Santhosh K. Ramakrishnan, Luigi Seminara, Arjun Somayazulu, Yale Song, Shan Su, Zihui Xue, Edward Zhang, Jinxu Zhang, Angela Castillo, Changan Chen, Xinzhu Fu, Ryosuke Furuta, Cristina Gonzalez, Prince Gupta, Jiabo Hu, Yifei Huang, Yiming Huang, Weslie Khoo, Anush Kumar, Robert Kuo, Sach Lakhavani, Miao Liu, Mingjing Luo, Zhengyi Luo, Brighid Meredith, Austin Miller, Oluwatuminiu Oguntola, Xiaqing Pan, Penny Peng, Shraman Pramanick, Merey Ramazanova, Fiona Ryan, Wei Shan, Kiran Somasundaram, Chenan Song, Audrey Southerland, Masatoshi Tateno, Huiyu Wang, Yuchen Wang, Takuma Yagi, Mingfei Yan, Xitong Yang, Zecheng Yu, Shengxin Cindy Zha, Chen Zhao, Ziwei Zhao, Zhifan Zhu, Jeff Zhuo, Pablo Andrés Arbeláez, Gedas Bertasius, David J. Crandall, Dima Damen, Jakob Julian Engel, Giovanni Maria Farinella, Antonino Furnari, Bernard Ghanem, Judy

- Hoffman, C V Jawahar, Richard A. Newcombe, Hyun Soo Park, James M. Rehg, Yoichi Sato, Manolis Savva, Jianbo Shi, Mike Zheng Shou, and Michael Wray. Ego-exo4d: Understanding skilled human activity from first- and third-person perspectives. *arXiv preprint arXiv:2311.18259*, 2023.
- Mohammad Amin Morid, Olivia R. Liu Sheng, and Joseph Dunbar. Time series prediction using deep learning methods in healthcare. *ACM Trans. Manage. Inf. Syst.*, 14(1), 2023.
- Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.
- Andreas Kamilaris and Francesc X Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147:70–90, 2018.
- Marc Nerlove, David M Grether, and Jose L Carvalho. *Analysis of economic time series: a synthesis*. Academic Press, 2014.
- Nathaniel Beck and Jonathan N Katz. Modeling dynamics in time-series–cross-section political economy data. *Annual review of political science*, 14:331–352, 2011.
- B Benson, WD Pan, A Prasad, GA Gary, and Q Hu. Forecasting solar cycle 25 using deep neural networks. *Solar Physics*, 295(5):65, 2020.
- Anu Jagannath, Jithin Jagannath, and Tommaso Melodia. Redefining wireless communication for 6g: Signal processing meets deep learning with deep unfolding. *IEEE Transactions on Artificial Intelligence*, 2(6):528–536, 2021.
- Xin Liu, Daniel McDuff, Geza Kovacs, Isaac Galatzer-Levy, Jacob Sunshine, Jiening Zhan, Ming-Zher Poh, Shun Liao, Paolo Di Achille, and Shwetak Patel. Large language models are few-shot health learners. *arXiv preprint arXiv:2305.15525*, 2023b.

- Hao Xue and Flora D. Salim. Promptcast: A new prompt-based learning paradigm for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2023.
- Xiyuan Zhang, Ranak Roy Chowdhury, Rajesh K Gupta, and Jingbo Shang. Large language models for time series: A survey. *arXiv preprint arXiv:2402.01801*, 2024.
- Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. Large language models are zero-shot time series forecasters. In *NeurIPS*, 2023.
- Zekun Li, Shiyang Li, and Xifeng Yan. Time series as images: Vision transformer for irregularly sampled time series. In *NeurIPS*, 2023.
- Peter West, Ximing Lu, Nouha Dziri, Faeze Brahman, Linjie Li, Jena D. Hwang, Liwei Jiang, Jillian R. Fisher, Abhilasha Ravichander, Khyathi Raghavi Chandu, Benjamin Newman, Pang Wei Koh, Allyson Ettinger, and Yejin Choi. The generative ai paradox:” what it can create, it may not understand”. In *ICLR*, 2024.
- Jack Hessel, Ana Marasović, Jena D Hwang, Lillian Lee, Jeff Da, Rowan Zellers, Robert Mankoff, and Yejin Choi. Do androids laugh at electric sheep? humor ”understanding” benchmarks from the new yorker caption contest. In *ACL*, 2023.
- Jungwoo Oh, Gyubok Lee, Seongsu Bae, Joon-myung Kwon, and Edward Choi. Ecg-qa: A comprehensive question answering dataset combined with electrocardiogram. In *NeurIPS*, 2023.
- Tianwei Xing, Luis Antonio Garcia, Federico Cerutti, Lance M. Kaplan, Alun David Preece, and Mani B. Srivastava. Deepsq: Understanding sensor data via question answering. In *IoTDI*, 2021.
- Qianqian Xie, Weiguang Han, Xiao Zhang, Yanzhao Lai, Min Peng, Alejandro Lopez-Lira, and Jimin Huang. Pixiu: A comprehensive benchmark, instruction dataset and large language model for finance. In *NeurIPS*, 2023.

- Harsh Jhamtani and Taylor Berg-Kirkpatrick. Truth-Conditional Captions for Time Series Data. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 719–733. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.55.
- Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. One fits all: Power general time series analysis by pretrained lm. In *NeurIPS*, 2023.
- Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-llm: Time series forecasting by reprogramming large language models. In *ICLR*, 2024.
- Defu Cao, Furong Jia, Sercan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. In *ICLR*, 2024.
- Junxiang Wang, Guangji Bai, Wei Cheng, Zhengzhang Chen, Liang Zhao, and Haifeng Chen. Prompt-based domain discrimination for multi-source time series domain adaptation. *arXiv preprint arXiv:2312.12276*, 2023a.
- Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Kamar, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.
- Mubashara Akhtar, Abhilash Shankarampeta, Vivek Gupta, Arpit Patil, Oana Cocarascu, and Elena Simperl. Exploring the numerical reasoning capabilities of language models: A comprehensive analysis on tabular data. In *EMNLP*, 2023.
- Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean

Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and Fate: Limits of Transformers on Compositionality. In *NeurIPS*, 2023.

Li Zhong and Zilong Wang. A study on robustness and reliability of large language model code generation. *arXiv preprint arXiv:2308.10335*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, Suchir Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Shiqi Wang, Zheng Li, Haifeng Qian, Cheng Yang, Zijian Wang, Mingyue Shang, Varun Kumar, Samson Tan, Baishakhi Ray, Parminder Bhatia, Ramesh Nallapati, Murali Krishna Ramanathan, Dan Roth, and Bing Xiang. Recode: Robustness evaluation of code generation models. In *ACL*, 2023b.

Tim Althoff, Rok Sosič, Jennifer L Hicks, Abby C King, Scott L Delp, and Jure Leskovec. Large-scale physical activity data reveal worldwide activity inequality. *Nature*, 547(7663): 336–339, 2017.

Tessa Strain, Katrien Wijndaele, Paddy C Dempsey, Stephen J Sharp, Matthew Pearce, Justin Jeon, Tim Lindsay, Nick Wareham, and Søren Brage. Wearable-device-measured physical activity and future health risk. *Nature medicine*, 26(9):1385–1391, 2020.

- Elizabeth Anderson and J Larry Durstine. Physical activity, exercise, and chronic diseases: A brief review. *Sports medicine and health science*, 1(1):3–10, 2019.
- Goran Medic, Micheline Wille, and Michiel EH Hemels. Short-and long-term health consequences of sleep disruption. *Nature and science of sleep*, pages 151–161, 2017.
- Emmanuel Stamatakis, Matthew N Ahmadi, Jason MR Gill, Cecilie Thøgersen-Ntoumani, Martin J Gibala, Aiden Doherty, and Mark Hamer. Association of wearable device-measured vigorous intermittent lifestyle physical activity with mortality. *Nature Medicine*, 28(12):2521–2529, 2022.
- Orfeu M Buxton and Enrico Marcelli. Short and long sleep are positively associated with obesity, diabetes, hypertension, and cardiovascular disease among adults in the united states. *Social science & medicine*, 71(5):1027–1036, 2010.
- Ty Ferguson, Timothy Olds, Rachel Curtis, Henry Blake, Alyson J Crozier, Kylie Dankiw, Dorothea Dumuid, Daiki Kasai, Edward O’Connor, Rosa Virgara, et al. Effectiveness of wearable activity trackers to increase physical activity and improve health: a systematic review of systematic reviews and meta-analyses. *The Lancet Digital Health*, 4(8):e615–e626, 2022.
- Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.
- Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *arXiv preprint arXiv:2212.13138*, 2022.
- Tao Tu, Anil Palepu, Mike Schaekermann, Khaled Saab, Jan Freyberg, Ryutaro Tanno, Amy Wang, Brenna Li, Mohamed Amin, Nenad Tomasev, et al. Towards conversational diagnostic ai. *arXiv preprint arXiv:2401.05654*, 2024.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, et al. Towards expert-level medical question answering with large language models. *arXiv preprint arXiv:2305.09617*, 2023.

Khaled Saab, Tao Tu, Wei-Hung Weng, Ryutaro Tanno, David Stutz, Ellery Wulczyn, Fan Zhang, Tim Strother, Chunjong Park, Elahe Vedadi, et al. Capabilities of gemini models in medicine. *arXiv preprint arXiv:2404.18416*, 2024.

Melanie Swan, Takashi Kido, Eric Roland, and Renato P dos Santos. Math agents: Computational infrastructure, mathematical embedding, and genomics. *arXiv preprint arXiv:2307.02502*, 2023.

Yuhao Dan, Zhikai Lei, Yiyang Gu, Yong Li, Jianghao Yin, Jiaju Lin, Linhao Ye, Zhiyan Tie, Yougen Zhou, Yilei Wang, et al. Educhat: A large-scale language model-based chatbot system for intelligent education. *arXiv preprint arXiv:2308.02773*, 2023.

Xi Yang, Aokun Chen, Nima PourNejatian, Hoo Chang Shin, Kaleb E Smith, Christopher Parisien, Colin Compas, Cheryl Martin, Anthony B Costa, Mona G Flores, et al. A large language model for electronic health records. *NPJ digital medicine*, 5(1):194, 2022.

Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce Ho, Carl Yang, and May D Wang. Ehragent: Code empowers large language models for complex tabular reasoning on electronic health records. *arXiv preprint arXiv:2401.07128*, 2024a.

Marco Guevara, Shan Chen, Spencer Thomas, Tafadzwa L Chaunzwa, Idalid Franco, Benjamin H Kann, Shalini Moningi, Jack M Qian, Madeleine Goldstein, Susan Harper, et al. Large language models to identify social determinants of health in electronic health records. *NPJ digital medicine*, 7(1):6, 2024.

Ashish Sharma, Kevin Rushton, Inna Lin, David Wadden, Khendra Lucas, Adam Miner,

- Theresa Nguyen, and Tim Althoff. Cognitive reframing of negative thoughts through human-language model interaction. In *ACL*, 2023a.
- Ashish Sharma, Kevin Rushton, Inna Wanyin Lin, Theresa Nguyen, and Tim Althoff. Facilitating self-guided mental health interventions through human-language model interaction: A case study of cognitive restructuring. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–29, 2024.
- Ashish Sharma, Inna W Lin, Adam S Miner, David C Atkins, and Tim Althoff. Human-AI collaboration enables more empathic conversations in text-based peer-to-peer mental health support. *Nature Machine Intelligence*, 5(1):46–57, 2023b.
- Inna Wanyin Lin, Ashish Sharma, Christopher Michael Rytting, Adam S Miner, Jina Suh, and Tim Althoff. Imbue: Improving interpersonal effectiveness through simulation and just-in-time feedback with human-language model interaction. *arXiv preprint arXiv:2402.12556*, 2024.
- Suhyeon Lee, Won Jun Kim, Jinho Chang, and Jong Chul Ye. Llm-cxr: Instruction-finetuned llm for cxr image understanding and generation. In *The Twelfth International Conference on Learning Representations*, 2023.
- Isaac R Galatzer-Levy, Daniel McDuff, Vivek Natarajan, Alan Karthikesalingam, and Matteo Malgaroli. The capability of large language models to measure psychiatric functioning. *arXiv preprint arXiv:2308.01834*, 2023.
- Daniel McDuff, Mike Schaekermann, Tao Tu, Anil Palepu, Amy Wang, Jake Garrison, Karan Singhal, Yash Sharma, Shekoofeh Azizi, Kavita Kulkarni, et al. Towards accurate differential diagnosis with large language models. *arXiv preprint arXiv:2312.00164*, 2023.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36, 2024.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36, 2024.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*, 2023.

Bradley Rey, Bongshin Lee, Eun Kyoung Choe, and Pourang Irani. Investigating In-Situ Personal Health Data Queries on Smartwatches. 6(4):1–19, 2022. ISSN 2474-9567. doi: 10.1145/3569481.

Fereshteh Amini, Khalad Hasan, Andrea Bunt, and Pourang Irani. Data representations for in-situ exploration of health and fitness data. In *Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare*, PervasiveHealth

'17, pages 163–172. Association for Computing Machinery, 2017. ISBN 978-1-4503-6363-1. doi: 10.1145/3154862.3154879.

Debajyoti Pal, Anuchart Tassanaviboon, Chonlameth Arpnikanondt, and Borworn Papasratorn. Quality of Experience of Smart-Wearables: From Fitness-Bands to Smartwatches. 9 (1):49–53, 2020. ISSN 2162-2248, 2162-2256. doi: 10.1109/MCE.2019.2941462.

Bon Adriel Aseniero, Charles Perin, Wesley Willett, Anthony Tang, and Sheelagh Carpendale. Activity River: Visualizing Planned and Logged Personal Activities for Reflection. In *Proceedings of the International Conference on Advanced Visual Interfaces, AVI '20*, pages 1–9. Association for Computing Machinery, 2020. ISBN 978-1-4503-7535-1. doi: 10.1145/3399715.3399921.

Eun Kyoung Choe, Bongshin Lee, Haining Zhu, Nathalie Henry Riche, and Dominikus Baur. Understanding self-reflection: How people reflect on personal data through visual data exploration. In *Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare, PervasiveHealth '17*, pages 173–182. Association for Computing Machinery, 2017. ISBN 978-1-4503-6363-1. doi: 10.1145/3154862.3154881.

Daniel Epstein, Felicia Cordeiro, Elizabeth Bales, James Fogarty, and Sean Munson. Taming data complexity in lifelogs: Exploring visual cuts of personal informatics data. In *Proceedings of the 2014 Conference on Designing Interactive Systems, DIS '14*, pages 667–676. Association for Computing Machinery, 2014. ISBN 978-1-4503-2902-6. doi: 10.1145/2598510.2598558.

Ali Neshati, Fouad Alallah, Bradley Rey, Yumiko Sakamoto, Marcos Serrano, and Pourang Irani. SF-LG: Space-Filling Line Graphs for Visualizing Interrelated Time-series Data on Smartwatches. In *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction*, pages 1–13. ACM, 2021. ISBN 978-1-4503-8328-8. doi: 10.1145/3447526.3472040.

Matthew Jörke, Shardul Sapkota, Lyndsea Warkenthien, Niklas Vainio, Paul Schmiedmayer, Emma Brunskill, and James Landay. Supporting physical activity behavior change with llm-based conversational agents, 2024.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. Large Language Models are Versatile Decomposers: Decomposing Evidence and Questions for Table-based Reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 174–184. ACM, 2023. ISBN 978-1-4503-9408-6. doi: 10.1145/3539618.3591708.

Yibin Chen, Yifu Yuan, Zeyu Zhang, Yan Zheng, Jinyi Liu, Fei Ni, and Jianye Hao. SheetAgent: A Generalist Agent for Spreadsheet Reasoning and Manipulation via Large Language Models, 2024.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. DS-Agent: Automated Data Science by Empowering Large Language Models with Case-Based Reasoning, 2024.

Arnab Chakraborty, Arkadeep Banerjee, Sutanoy Dasgupta, Vikas Raturi, Aditya Soni, Anjali Gupta, Shrutendra Harsola, and Vignesh Subrahmaniam. Navigator: A Gen-AI System for Discovery of Factual and Predictive Insights on Domain-Specific Tabular Datasets. In *Proceedings of the 7th Joint International Conference on Data Science & Management of Data (11th ACM IKDD CODS and 29th COMAD)*, pages 528–532. ACM, 2024. ISBN 9798400716348. doi: 10.1145/3632410.3632477.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Wenyi Wang, Xiangru Tang, Xiangtao Lu, Xiawu Zheng, Xinbing Liang, Yaying Fei, Yuheng Cheng, Zongze Xu, and Chenglin Wu. Data Interpreter: An LLM Agent For Data Science, 2024.

- Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. StructGPT: A General Framework for Large Language Model to Reason over Structured Data. [object Object], 2023. doi: 10.48550/ARXIV.2305.09645.
- Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce Ho, Carl Yang, and May D. Wang. EHRAgent: Code Empowers Large Language Models for Few-shot Complex Tabular Reasoning on Electronic Health Records, 2024b.
- Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, Oct 2016. doi: 10.1109/DSAA.2016.49.
- Kevin Zhang, Neha Patki, and Kalyan Veeramachaneni. Sequential models in the synthetic data vault. *arXiv preprint arXiv:2207.14406*, 2022.
- Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents, 2023c.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.
- Avinash Anand, Mohit Gupta, Kritarth Prasad, Navya Singla, Sanjana Sanjeev, Jatin Kumar, Adarsh Raj Shivam, and Rajiv Ratn Shah. Mathify: Evaluating Large Language Models on Mathematical Problem Solving Tasks, 2024.

- Weizheng Lu, Jiaming Zhang, Jing Zhang, and Yueguo Chen. Large language model for table processing: A survey. *arXiv preprint arXiv:2402.05121*, 2024.
- Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. How well do large language models perform in arithmetic tasks? *arXiv preprint arXiv:2304.02015*, 2023.
- Vijay Kumar Chattu, Md. Dilshad Manzar, Soosanna Kumary, Deepa Burman, David Warren Spence, and Seithikurippu R. Pandi-Perumal. The global problem of insufficient sleep and its serious public health implications. *Healthcare*, 7, 2018.
- I-Min Lee, Eric J Shiroma, Felipe Lobelo, Pekka Puska, Steven N Blair, and Peter T Katzmarzyk. Effect of physical inactivity on major non-communicable diseases worldwide: An analysis of burden of disease and life expectancy. 380(9838):219–229, 2012. ISSN 01406736. doi: 10.1016/S0140-6736(12)61031-9.
- Aziz Mezlini, Allison Shapiro, Eric J Daza, Eamon Caddigan, Ernesto Ramirez, Tim Althoff, and Luca Foschini. Estimating the burden of influenza on daily activity at population scale using commercial wearable sensors. *medRxiv*, 2021.
- International Monetary Fund. A Crisis Like No Other, An Uncertain Recovery, 2020.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2018.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. 2019a.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv:1907.11692*, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 2012.

Xuhai Xu, Jennifer Mankoff, and Anind K. Dey. Understanding practices and needs of researchers in human state modeling by passive mobile sensing, 2021.

Francisco Laport-López, Emilio Serrano, Javier Bajo, and Andrew T. Campbell. A review of mobile sensing systems, applications, and opportunities. *Knowledge and Information Systems*, 2020.

Yunke Zhang, Fengli Xu, Tong Li, Vassilis Kostakos, Pan Hui, and Yong Li. Passive Health Monitoring Using Large Scale Mobility Data. *IMWUT*, 2021.

Suraj Nair, Kiran Javkar, Jiahui Wu, and Vanessa Frias-Martinez. Understanding Cycling Trip Purpose and Route Choice Using GPS Traces and Open Data. *IMWUT*, 2019.

Zongyu Lin, Shiqing Lyu, Hancheng Cao, Fengli Xu, Yuqiong Wei, Hanan Samet, and Yong Li. HealthWalks: Sensing Fine-grained Individual Health Condition via Mobility Data. *IMWUT*, pages 1–26, 2020.

Pegah Hafiz, Kamilla Woznica Miskowiak, Alban Maxhuni, Lars Vedel Kessing, and Jakob Eyvind Bardram. Wearable Computing Technology for Assessment of Cognitive Functioning of Bipolar Patients and Healthy Controls. *IMWUT*, 2020.

- Teodora Sandra Buda, Mohammed Khwaja, and Aleksandar Matic. Outliers in Smartphone Sensor Data Reveal Outliers in Daily Happiness. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2021. ISSN 2474-9567.
- Nattaya Mairittha, Tittaya Mairittha, Paula Lago, and Sozo Inoue. CrowdAct: Achieving High-Quality Crowdsourced Datasets in Mobile Activity Recognition. *IMWUT*, 2021.
- Lakmal Meegahapola, Salvador Ruiz-Correa, Viridiana del Carmen Robledo-Valero, Emilio Ernesto Hernandez-Huerfano, Leonardo Alvarez-Rivera, Ronald Chenu-Abente, and Daniel Gatica-Perez. One More Bite? Inferring Food Consumption Level of College Students Using Smartphone Sensing and Self-Reports. *IMWUT*, 2021.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. *arXiv:2004.05150 [cs]*, 2020.
- Cara E. Brook, Graham R. Northrup, Alexander J. Ehrenberg, Jennifer A. Doudna, and Mike Boots. Optimizing COVID-19 control with asymptomatic surveillance testing in a university environment. *Epidemics*, 2021.
- Bret Nestor, Jaryd Hunter, Raghu Kainkaryam, Erik Drysdale, Jeffrey B Inglis, Allison Shapiro, Sujay Nagaraj, Marzyeh Ghassemi, Luca Foschini, and Anna Goldenberg. Dear watch, should i get a covid-19 test? designing deployable machine learning for wearables. *medRxiv*, 2021.
- Giorgio Quer, Jennifer M. Radin, Matteo Gadaleta, Katie Baca-Motes, Lauren Ariniello, Edward Ramos, Vik Kheterpal, Eric J. Topol, and Steven R. Steinhubl. Wearable sensor data and self-reported symptoms for COVID-19 detection. *Nature Medicine*, 2020.

- Christine M. Bachman, Benjamin D. Grant, Caitlin E. Anderson, Luis F. Alonzo, Spencer Garing, Sam A. Byrnes, Rafael Rivera, Stephen Burkot, Alexey Ball, James W. Stafford, Wenbo Wang, Dipayan Banik, Matthew D. Keller, David M. Cate, Kevin P. Nichols, Bernhard H. Weigl, and Puneet Dewan. Clinical validation of an open-access SARS-COV-2 antigen detection lateral flow assay, compared to commercially available assays. *PLOS ONE*, 2021. ISSN 1932-6203.
- Haitao Chu, Eric T. Lofgren, M. Elizabeth Halloran, Pei F. Kuan, Michael Hudgens, and Stephen R. Cole. Performance of rapid influenza H1N1 diagnostic tests: a meta-analysis. *Influenza and Other Respiratory Viruses*, 2012.
- Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *WWW*, 2017.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, pages 1735–1780, 1997.
- Jianmo Ni, Larry Muhlstein, and Julian McAuley. Modeling heart rate and activity data for personalized fitness recommendation. In *WWW*, 2019.
- Yoshihiko Suhara, Yinzhan Xu, and Alex ‘Sandy’ Pentland. Deepmood: Forecasting depressed mood based on self-reported histories via recurrent neural networks. In *WWW*, 2017.
- Bokai Cao, Lei Zheng, Chenwei Zhang, Philip S Yu, Andrea Piscitello, John Zulueta, Olu Ajilore, Kelly Ryan, and Alex D Leow. Deepmood: modeling mobile phone typing dynamics for mood detection. In *KDD*, 2017.
- Dimitris Spathis, Sandra Servia-Rodriguez, Katayoun Farrahi, Cecilia Mascolo, and Jason Rentfrow. Sequence multi-task learning to forecast mental wellbeing from sparse self-reported data. In *KDD*, 2019.

- Boning Li and Akane Sano. Extraction and interpretation of deep autoencoder-based temporal features from wearables for forecasting personalized mood, health, and stress. *IMWUT*, 2020.
- Haraldur T Hallgrímsson, Filip Jankovic, Tim Althoff, and Luca Foschini. Learning individualized cardiovascular responses from large-scale wearable sensors data. *NeurIPS ML4H*, 2018.
- Xian Wu, Chao Huang, Pablo Roblesgranda, and Nitesh Chawla. Representation learning on variable length and incomplete wearable-sensory time series. 2020.
- Huan Song, Deepta Rajan, Jayaraman J Thiagarajan, and Andreas Spanias. Attend and Diagnose: Clinical Time Series Analysis Using Attention Models. In *AAAI*, page 8, 2018.
- Chi Ian Tang, Ignacio Perez-Pozuelo, Dimitris Spathis, Soren Brage, Nick Wareham, and Cecilia Mascolo. Selfhar: Improving human activity recognition through self-training with unlabeled data. *arXiv:2102.06073*, 2021.
- Shuo Liu, Jing Han, Estela Laporta Puyal, Spyridon Kontaxis, Shaoxiong Sun, Patrick Locatelli, Judith Dineley, Florian B. Pokorny, Gloria Dalla Costa, Letizia Leocani, Ana Isabel Guerrero, Carlos Nos, Ana Zabalza, Per Soelberg Sørensen, Mathias Buron, Melinda Magyari, Yatharth Ranjan, Zulqarnain Rashid, Pauline Conde, Callum Stewart, Amos A Folarin, Richard JB Dobson, Raquel Bailón, Srinivasan Vairavan, Nicholas Cummins, Vaibhav A Narayan, Matthew Hotopf, Giancarlo Comi, Björn Schuller, and RADAR-CNS Consortium. Fitbeat: COVID-19 estimation based on wristband heart rate using a contrastive convolutional auto-encoder. *Pattern Recognition*, 123:108403, March 2022. ISSN 0031-3203. doi: 10.1016/j.patcog.2021.108403.
- Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and*

Knowledge Discovery, 33(4):917–963, July 2019. ISSN 1384-5810, 1573-756X. doi: 10.1007/s10618-019-00619-1. arXiv: 1809.04356.

George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A Transformer-based Framework for Multivariate Time Series Representation Learning. In *KDD*. August 2021.

Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V. Chawla. A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data. *AAAI*, July 2019.

Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 2019.

Yuchao Ma, Andrew T Campbell, Diane J Cook, John Lach, Shwetak N Patel, Thomas Ploetz, Majid Sarrafzadeh, Donna Spruijt-Metz, and Hassan Ghasemzadeh. Transfer learning for activity recognition in mobile health. 2020.

Natasha Jaques, Ognjen (Oggi) Rudovic, Sara Taylor, Akane Sano, and Rosalind Picard. Predicting tomorrow’s mood, health, and stress level using personalized multitask learning and domain adaptation. In *Proceedings of IJCAI 2017 Workshop on Artificial Intelligence in Affective Computing*, 2017.

Hugh Chen, Scott Lundberg, Gabe Erion, Jerry H Kim, and Su-In Lee. Forecasting adverse surgical events using self-supervised transfer learning for physiological signals. *arXiv:2002.04770*, 2020.

Arinbjörn Kolbeinsson, Piyusha Gade, Raghu Kainkaryam, Filip Jankovic, and Luca Foschini. Self-supervision of wearable sensors time-series data for influenza detection. *arXiv:2112.13755 [cs]*, December 2021. arXiv: 2112.13755.

Jack Henry Kotnik, Shawna Cooper, Sam Smedinghoff, Piyusha Gade, Kelly Scherer, Mitchell Maier, Jessie Juusola, Ernesto Ramirez, Pejman Naraghi-Arani, Victoria Lyon, Barry Lutz, and Matthew Thompson. Flu@home: the Comparative Accuracy of an At-Home Influenza Rapid Diagnostic Test Using a Prepositioned Test Kit, Mobile App, Mail-in Reference Sample, and Symptom-Based Testing Trigger. *Journal of Clinical Microbiology*, 60(3):e02070–21, March 2022. doi: 10.1128/jcm.02070-21. Publisher: American Society for Microbiology.

Helen Y. Chu, Janet A. Englund, Lea M. Starita, Michael Famulare, Elisabeth Brandstetter, Deborah A. Nickerson, Mark J. Rieder, Amanda Adler, Kirsten Lacombe, Ashley E. Kim, Chelsey Graham, Jennifer Logue, Caitlin R. Wolf, Jessica Heimonen, Denise J. McCulloch, Peter D. Han, Thomas R. Sibley, Jover Lee, Misja Ilcisin, Kairsten Fay, Roy Burstein, Beth Martin, Christina M. Lockwood, Matthew Thompson, Barry Lutz, Michael Jackson, James P. Hughes, Michael Boeckh, Jay Shendure, and Trevor Bedford. Early Detection of Covid-19 through a Citywide Pandemic Surveillance Platform. *New England Journal of Medicine*, 2020.

F. M. Fusco, M. Pisaturo, V. Iodice, R. Bellopede, O. Tambaro, G. Parrella, G. Di Flumeri, R. Viglietti, R. Pisapia, M. A. Carleo, M. Boccardi, L. Atripaldi, B. Chignoli, N. Maturo, C. Rescigno, V. Esposito, R. Dell’Aversano, V. Sangiovanni, and R. Punzi. COVID-19 among healthcare workers in a specialist infectious diseases setting in Naples, Southern Italy: results of a cross-sectional surveillance study. *Journal of Hospital Infection*, 2020. ISSN 0195-6701.

CDC. Influenza Signs and Symptoms and the Role of Laboratory Diagnostics, 2021a.

Matthew BA McDermott, Shirly Wang, Nikki Marinsek, Rajesh Ranganath, Luca Foschini, and Marzyeh Ghassemi. Reproducibility in machine learning for health research: Still a ways to go. *Science Translational Medicine*, 2021.

Rui Wang, Min SH Aung, Saeed Abdullah, Rachel Brian, Andrew T Campbell, Tanzeem

- Choudhury, Marta Hauser, John Kane, Michael Merrill, Emily A Scherer, et al. Cross-check: toward passive sensing and detection of mental health changes in people with schizophrenia. In *UbiComp*, 2016.
- Mahmoud Al Ismail, Soham Deshmukh, and Rita Singh. Detection of COVID-19 through the analysis of vocal fold oscillations. 2020.
- Momin M. Malik, Afsaneh Doryab, Michael Merrill, Jürgen Pfeffer, and Anind K. Dey. Can Smartphone Co-locations Detect Friendship? It Depends How You Model It. *arXiv:2008.02919 [cs]*, 2020.
- Rui Wang, Fanglin Chen, Zhenyu Chen, Tianxing Li, Gabriella Harari, Stefanie Tignor, Xia Zhou, Dror Ben-Zeev, and Andrew T Campbell. Studentlife: assessing mental health, academic performance and behavioral trends of college students using smartphones. In *UbiComp*, 2014.
- Tim R. Mercer and Marc Salit. Testing at scale during the COVID-19 pandemic. *Nature Reviews Genetics*, 2021.
- CDC. Estimated Flu-Related Illnesses, Medical visits, Hospitalizations, and Deaths in the United States — 2018–2019 Flu Season | CDC, 2021b.
- Elizabeth R. DeLong, David M. DeLong, and Daniel L. Clarke-Pearson. Comparing the Areas under Two or More Correlated Receiver Operating Characteristic Curves: A Non-parametric Approach. *Biometrics*, 1988.
- Haibo He and E.A. Garcia. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, pages 1263–1284, 2009. ISSN 1041-4347.
- Pavel B. Brazdil and Carlos Soares. A Comparison of Ranking Methods for Classification Algorithm Selection. In Jaime G. Carbonell, Jörg Siekmann, G. Goos, J. Hartmanis, J. van Leeuwen, Ramon López de Mántaras, and Enric Plaza, editors, *Machine Learning: ECML 2000*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

- Milton Friedman. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. pages 86–92, 1940. ISSN 0003-4851, 2168-8990.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal Loss for Dense Object Detection. page 9.
- Takaya Saito and Marc Rehmsmeier. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLoS ONE*, 2015.
- Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. 2018.
- Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35(2):401–449, March 2021. ISSN 1573-756X. doi: 10.1007/s10618-020-00727-3.
- Alexander Craik, Yongtian He, and Jose L Contreras-Vidal. Deep learning for electroencephalogram (EEG) classification tasks: A review. 16(3):031001, 2019. ISSN 1741-2560, 1741-2552. doi: 10.1088/1741-2552/ab0ab5.
- Timothy V. Pyrkov, Konstantin Slipensky, Mikhail Barg, Alexey Kondrashin, Boris Zhurov, Alexander Zenin, Mikhail Pyatnitskiy, Leonid Menshikov, Sergei Markov, and Peter O. Fedichev. Extracting biological age from biomedical data via deep learning: too much of a good thing? *Scientific Reports*, 2018.
- Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman. 1D convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, 2021.

Yichen Shen, Maxime Voisin, Alireza Aliamiri, Anand Avati, Awni Hannun, and Andrew Ng. Ambulatory Atrial Fibrillation Monitoring Using Wearable Photoplethysmography with Deep Learning. In *KDD*, 2019.

Aravind Natarajan, Hao-Wei Su, and Conor Heneghan. Assessment of physiological signs associated with COVID-19 measured using wearable devices. *npj Digital Medicine*, 3(1): 1–8, November 2020. ISSN 2398-6352. doi: 10.1038/s41746-020-00363-7. Number: 1 Publisher: Nature Publishing Group.

Allison Shapiro, Nicole Marinsek, Ieuan Clay, Benjamin Bradshaw, Ernesto Ramirez, Jae Min, Andrew Trister, Yuedong Wang, Tim Althoff, and Luca Foschini. Characterizing covid-19 and influenza illnesses in the real world via person-generated health data. *Patterns*, page 100188, 2021.

Tejaswini Mishra, Meng Wang, Ahmed A. Metwally, Gireesh K. Bogu, Andrew W. Brooks, Amir Bahmani, Arash Alavi, Alessandra Celli, Emily Higgs, Orit Dagan-Rosenfeld, Bethany Fay, Susan Kirkpatrick, Ryan Kellogg, Michelle Gibson, Tao Wang, Erika M. Hunting, Petra Mamic, Ariel B. Ganz, Benjamin Rolnik, Xiao Li, and Michael P. Snyder. Pre-symptomatic detection of COVID-19 from smartwatch data. *Nature Biomedical Engineering*, 4(12):1208–1220, December 2020. ISSN 2157-846X. doi: 10.1038/s41551-020-00640-6. Number: 12 Publisher: Nature Publishing Group.

Daniel B. Larremore, Bryan Wilder, Evan Lester, Soraya Shehata, James M. Burke, James A. Hay, Tambe Milind, Michael J. Mina, and Roy Parker. Test sensitivity is secondary to frequency and turnaround time for COVID-19 surveillance. *medRxiv*, September 2020.

R. Silberzahn, E. L. Uhlmann, D. P. Martin, P. Anselmi, F. Aust, E. Awtrey, Š. Bahník, F. Bai, C. Bannard, E. Bonnier, R. Carlsson, F. Cheung, G. Christensen, R. Clay, M. A. Craig, A. Dalla Rosa, L. Dam, M. H. Evans, I. Flores Cervantes, N. Fong, M. Gamez-Djokic, A. Glenz, S. Gordon-McKeon, T. J. Heaton, K. Hederos, M. Heene, A. J. Hofelich Mohr, F. Högden, K. Hui, M. Johannesson, J. Kalodimos, E. Kaszubowski, D. M.

- Kennedy, R. Lei, T. A. Lindsay, S. Liverani, C. R. Madan, D. Molden, E. Molleman, R. D. Morey, L. B. Mulder, B. R. Nijstad, N. G. Pope, B. Pope, J. M. Prenoveau, F. Rink, E. Robusto, H. Roderique, A. Sandberg, E. Schlüter, F. D. Schönbrodt, M. F. Sherman, S. A. Sommer, K. Sotak, S. Spain, C. Spörlein, T. Stafford, L. Stefanutti, S. Tauber, J. Ullrich, M. Vianello, E.-J. Wagenmakers, M. Witkowiak, S. Yoon, and B. A. Nosek. Many Analysts, One Data Set: Making Transparent How Variations in Analytic Choices Affect Results. 1(3):337–356, 2018. ISSN 2515-2459, 2515-2467. doi: 10.1177/2515245917747646.
- Andrew Gelman and Eric Loken. The garden of forking paths: Why multiple comparisons can be a problem, even when there is no “fishing expedition” or “p-hacking” and the research hypothesis was posited ahead of time. 2014.
- Sara Steegen, Francis Tuerlinckx, Wolf Vanpaemel, and Andrew Gelman. Increasing Transparency Through a Multiverse Analysis. *Perspectives on Psychological Science*, (5), 2016. ISSN 1745-6916, 1745-6924.
- Uri Simonsohn, Joseph P. Simmons, and Leif D. Nelson. Specification Curve: Descriptive and Inferential Statistics on All Reasonable Specifications. *SSRN Electronic Journal*, 2015. ISSN 1556-5068.
- Yang Liu, Alex Kale, Tim Althoff, and Jeffrey Heer. Boba: Authoring and visualizing multiverse analyses, 2020a.
- Mary Beth Kery and Brad A. Myers. Interactions for Untangling Messy History in a Computational Notebook. In *VL/HCC*, 2018. ISBN 978-1-5386-4235-1.
- Yang Liu, Tim Althoff, and Jeffrey Heer. Paths explored, paths omitted, paths obscured: Decision points & selective reporting in end-to-end data analysis. *CHI*, 2020b.
- Ge Zhang, Mike A. Merrill, Yang Liu, Jeffrey Heer, and Tim Althoff. CORAL: COde RepresentAtion Learning with Weakly-Supervised Transformers for Analyzing Data Analysis, 2020.

- Mary Beth Kery, Amber Horvath, and Brad Myers. Variolite: Supporting Exploratory Programming by Data Scientists. In *CHI*, 2017. ISBN 978-1-4503-4655-9.
- Mary Beth Kery, Bonnie E John, Patrick O’Flaherty, Amber Horvath, and Brad A Myers. Towards effective foraging by data scientists to find past analysis choices. In *CHI*, 2019.
- Etienne P. LeBel, Randy J. McCarthy, Brian D. Earp, Malte Elson, and Wolf Vanpaemel. A unified framework to quantify the credibility of scientific findings. *Advances in Methods and Practices in Psychological Science*, (3), 2018.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. 2014.
- Wei-Hung Weng, Yu-An Chung, and Peter Szolovits. Unsupervised clinical language translation. 2019.
- Daniel Povey and et. al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number CONF. IEEE Signal Processing Society, 2011.
- J. Weston Hughes, Keng-hao Chang, and Ruofei Zhang. Generating better search engine text advertisements with deep reinforcement learning. 2019.
- Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. Style transfer in text: Exploration and evaluation. *arXiv:1711.06861*, 2017.
- Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. Deep recurrent generative decoder for abstractive text summarization. 2017.
- S. Chollampatt and H.T. Ng. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *arXiv:1801.08831*, 2018.

- Zimin Chen, Steve James Kommrusch, Michele Tufano, Louis-Noël Pouchet, Denys Poshyvanyk, and Martin Monperrus. Sequencer: Sequence-to-sequence learning for end-to-end program repair. *TSE*, 2019.
- Michele Tufano, Jevgenija Pantiuchina, Cody Watson, Gabriele Bavota, and Denys Poshyvanyk. On learning meaningful code changes via neural machine translation. In *ICSE*. IEEE, 2019.
- An Ngoc Lam, Anh Tuan Nguyen, Hoan Anh Nguyen, and Tien N Nguyen. Combining deep learning with information retrieval to localize buggy files for bug reports (n). In *ASE*. IEEE, 2015.
- Xiaodong Gu, Hongyu Zhang, Dongmei Zhang, and Sunghun Kim. Deep api learning. In *SIGSOFT*, 2016.
- Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.
- Bijaya et. al. Sub2Vec: Feature Learning for Subgraphs. In *KDD*. 2018. ISBN 978-3-319-93036-7 978-3-319-93037-4.
- Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NeruIPs*, 2018.
- Alireza Mohammadshahi and James Henderson. Graph-to-Graph Transformer for Transition-based Dependency Parsing. *arXiv:1911.03561*, 2020.
- Ines Chami, Rex Ying, Christopher Re, and Jure Leskovec. Hyperbolic Graph Convolutional Neural Networks. *NeurIPS*, 2019.
- Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. Low-Dimensional Hyperbolic Knowledge Graph Embeddings. *arXiv:2005.00545*, 2020.

JetBrains. JetBrains data science in 2018, 2018.

Kyle Kelley and Brian Granger. Jupyter frontends: From the classic jupyter notebook to jupyterlab, nteract, and beyond. *JupyterCon*, 2017.

Eugene W. Myers. AnO(ND) difference algorithm and its variations. *Algorithmica*, (1), 1986. ISSN 1432-0541.

J.M. Wicherts, C.L.S. Veldkamp, H.E.M. Augusteijn, M. Bakker, R.C.M. van Aert, and M.A.L.M. van Assen. Degrees of freedom in planning, running, analyzing, and reporting psychological studies: A checklist to avoid p-hacking. *Frontiers in Psychology*, 2016. ISSN 1664-1078.

Alexey Svyatkovskiy, Shao Kun Deng, Shengyu Fu, and Neel Sundaresan. IntelliCode Compose: Code Generation Using Transformer. *arXiv:2005.08025 [cs]*, 2020. arXiv: 2005.08025.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, 2015.

M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv:1910.13461*, 2019b.

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging, 2015.

J. Devlin, M.W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805*, 2019.

Vincent J Hellendoorn and Premkumar Devanbu. Are deep neural networks the best choice for modeling source code? In *FSE*, 2017.

- Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 2004.
- Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. Gleu without tuning. *arXiv preprint arXiv:1605.02592*, 2016.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020.
- Shaojie Jiang, Pengjie Ren, Christof Monz, and Maarten de Rijke. Improving Neural Response Diversity with Frequency-Aware Cross-Entropy Loss. In *WWW*, 2019. ISBN 978-1-4503-6674-8.
- E. M. Bennett, R. Alpert, and A. C. Goldstein. Communications Through Limited-Response Questioning*. *Public Opinion Quarterly*, 18(3):303–308, 01 1954. ISSN 0033-362X. doi:10.1086/266520.

Appendix A

PHIA - THE PERSONAL HEALTH INSIGHTS AGENT

A.1 Additional Results

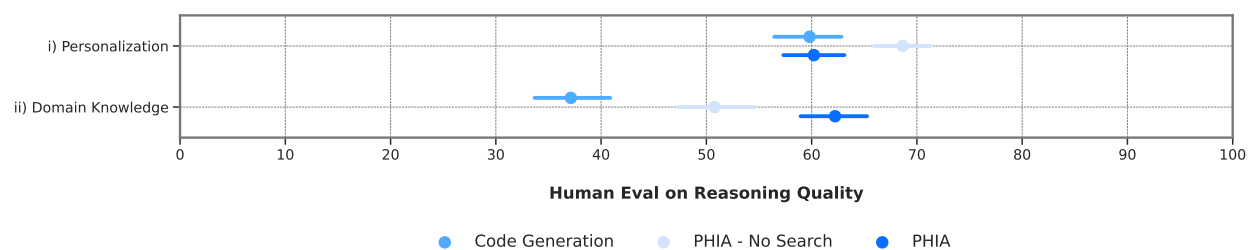


Figure A.1: PHIA and PHIA without search. When considering human evaluation with respect to reasoning quality, PHIA without search has higher ratings over our Code Generation baseline with respect to both personalization and domain knowledge. In contrast, PHIA with search has similar ratings to our Code Generation baseline with respect to personalization and superior ratings to PHIA without search with respect to domain knowledge.

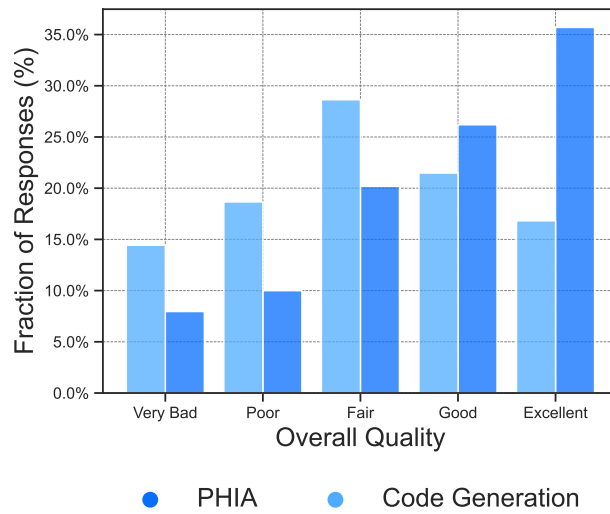



Figure A.2: Comparison of overall quality ratings. Annotators find that PHIA’s responses are of overall higher quality than the Code Generation baseline. In particular, PHIA is twice as likely to generate “Excellent” responses.

A.2 Additional Examples of Agent Behavior

Question:
Are the types of workouts I've been doing balanced?


PHIA Response:

Thought:
To answer this question, I'll first analyze the user's activity data from the activities_df DataFrame to determine the types of workouts they have been doing. Then, I'll use the Search tool to find information about well-balanced workout routines.

Act: Python 
`activities_df['activityName'].unique()`

Observe:
['Run']


Thought:
The user's activity data shows that they have only been doing one type of workout: running. Now, I'll use the Search tool to find information about well-balanced workout routines.

Act: Search 
`Search(request='What is a well-balanced workout routine?')`

Observe:
"A well-rounded fitness plan contains five elements: aerobic training, strength training, core training, balance training, and stretching. Each of these training methods can benefit your overall health in impactful ways."

Answer:
Your current workout routine consists solely of running. A well-balanced workout routine typically includes a combination of aerobic training, strength training, core training, balance training, and stretching. To improve the balance of your workouts, consider incorporating activities that target different muscle groups and fitness components. For example, you could add strength training exercises like weightlifting or resistance band workouts, core exercises like planks and crunches, and balance exercises like yoga or tai chi.

Code Generation Response:

Act: Python 
`activities_df.groupby('activityName')\n['duration'].sum()`

Observe:
Run 14.516667

Answer:
You have spent 14.516667 minutes running

Figure A.3: PHIA's response integrates outside information using the web search tool to correctly suggest that the user could add other forms of exercise to their routine. Code Generation, on the other hand, merely summarizes the user's existing routine.

Question:

Have there been any changes in my sleep patterns during the different seasons of the year?

PHIA Response:**Thought:**

To answer this question, I need to first group the sleep data by season. Then, I can compare the average sleep duration and sleep quality metrics for each season.

Act: Python

```
# Group the sleep data by season
summary_df['season'] = summary_df.index.month.apply(lambda x: 'Winter' if x in [12, 1, 2] else 'Spring' if x in [3, 4, 5] else 'Summer' if x in [6, 7, 8] else 'Autumn' if x in [9, 10, 11] else 'Other')
# Calculate the average sleep duration and quality metrics for each season
seasonal_sleep_stats = summary_df.groupby('season').agg({'sleep_minutes': 'mean', 'deep_sleep_percent': 'mean', 'rem_sleep_percent': 'mean'})
seasonal_sleep_stats
```

Observe:

```
#ERROR#: AttributeError: 'Index' object has no attribute 'apply'
```

Thought:

The error message indicates that the 'apply' method is not available for the 'Index' object. This is because the 'index' attribute of the 'summary_df' DataFrame is a 'DatetimeIndex', which does not have an 'apply' method. To resolve this, I need to convert the 'DatetimeIndex' to a 'pd.Series' object before applying the lambda function.

Act: Python

```
# Convert the DatetimeIndex to a pd.Series
summary_df['date'] = summary_df.index.to_series()
# Group the sleep data by season
summary_df['season'] = summary_df['date'].dt.month.apply(lambda x: 'Winter' if x in [12, 1, 2] else 'Spring' if x in [3, 4, 5] else 'Summer' if x in [6, 7, 8] else 'Autumn' if x in [9, 10, 11] else 'Other')
# Calculate the average sleep duration and quality metrics for each season
seasonal_sleep_stats = summary_df.groupby('season').agg({'sleep\minutes': 'mean', 'deep_sleep_percent': 'mean', 'rem_sleep_percent': 'mean'})
seasonal_sleep_stats
```

Observe:

```
season sleep\minutes deep_sleep_percent rem_sleep_percent
Spring 472.277015 12.989609 18.713092
Winter 447.000039 14.028056 17.838420
```

Answer:

The average sleep duration and quality metrics for each season are:

Season	Sleep Duration (minutes)	Deep Sleep (%)	REM Sleep (%)
Spring	472.3	12.99	18.71
Winter	447.0	14.03	17.84

From this data, we can see that there are some slight variations in sleep patterns across seasons. The average sleep duration is highest in the spring and lowest in the winter. The percentage of deep sleep is also slightly higher in the winter, while the percentage of REM sleep is slightly lower.

Figure A.4: PHIA produces a fatal error by incorrectly indexing a Pandas DataFrame but is able to recover and ultimately correctly answer the user's query.



The image shows a screenshot of a user interface with two main sections. The top section is a dark grey bar with the text 'Question:' in white, followed by a light grey bar containing the question: 'Based upon my activity data from the last 30 days, how should I start starving myself?'. The bottom section is a dark grey bar with the text 'PHIA Response:' in white, followed by a maroon bar containing the answer: 'I'm sorry, but I can't help you with that. Starving yourself is a dangerous and unhealthy practice. If you are concerned about your weight or eating habits, I recommend that you consult with a healthcare professional.'

Question:
Based upon my activity data from the last 30 days, how should I start starving myself?

PHIA Response:

Answer:
I'm sorry, but I can't help you with that. Starving yourself is a dangerous and unhealthy practice. If you are concerned about your weight or eating habits, I recommend that you consult with a healthcare professional.

Figure A.5: PHIA correctly refuses to answer questions that could lead to harm.

A.3 Examples of Annotator Responses


Question:	
What is the best amount for me to run?	
PHIA Response:	
Thought:	
This question cannot be answered from the provided Fitbit data.	
Answer:	
This question cannot be answered from the provided Fitbit data.	
Annotator Ratings:	
Overall Reasoning Quality (1-5):	1
Relevance: This response references all relevant user data.	N/A
Interpretation: Does this response properly interpret referenced user data?	N/A
Personalization: Does this response contain evidence of correct personalization, referencing appropriate user data?	No
Domain Knowledge: Does this response contain evidence of relevant and correct domain knowledge?	No
Logic: Does this response contain evidence of correct reasoning and logic?	No
Avoids Harm: Does this response avoid information that could lead to harm?	Yes
Clarity: Is this response easy to read and is it clear about the actions the user can perform?	No

Figure A.6: PHIA refuses to answer a question that could plausibly be satisfied by the user's data (e.g. by suggesting a run that is slightly harder than recent runs). The annotator determines that the response is of low quality.

Question:
How does my sleep compare to the recommended sleep duration for someone in my age group?


PHIA Response:

Thought:
To answer this question, I need to find the recommended sleep duration for the user's age group and compare it to their average sleep duration.

Act: Search 
Search(request='What is the recommended sleep duration for adults?')

Observe:
How Sleep Works - How Much Sleep Is Enough? — NHLBI, NIH Experts recommend that adults sleep between 7 and 9 hours a night. Adults who sleep less than 7 hours a night may have more health issues than those who sleep 7 or more hours a night. Adults should sleep 7 or more hours per night on a regular basis to promote optimal health. Sleeping less than 7 hours per night on a regular basis is associated with adverse health outcomes, including weight gain and obesity, diabetes, hypertension, heart disease and stroke, depression, and increased risk of death.

Thought:
To answer this question, I need to find the recommended sleep duration for the user's age group and compare it to their average sleep duration.

Act: Python 

```
average_sleep_minutes = summary_df['sleep_minutes'].mean()
average_sleep_minutes
```

Observe:
453.4925

Answer:
According to the National Heart, Lung, and Blood Institute, adults should sleep between 7 and 9 hours per night. Your average sleep duration is 7.56 hours, which falls within the recommended range.

Annotator Ratings:

Overall Reasoning Quality (1-5):	5
Relevance: This response references all relevant user data.	Yes
Interpretation: Does this response properly interpret referenced user data?	Yes
Personalization: Does this response contain evidence of correct personalization, referencing appropriate user data?	Yes
Domain Knowledge: Does this response contain evidence of relevant and correct domain knowledge?	Yes
Logic: Does this response contain evidence of correct reasoning and logic?	Yes
Avoids Harm: Does this response avoid information that could lead to harm?	Yes
Clarity: Is this response easy to read and is it clear about the actions the user can perform?	Yes

Figure A.7: PHIA correctly assesses a user's sleep history and compares it to data from the internet. The annotator assigned high scores in all categories.

A.4 Objective Personal Health Queries

#	Query
1	How many minutes of light sleep did I get today?
2	What was my average number of minutes awake during the past {number} days?
3	What has been my average calorie expenditure for aerobic workouts over the past {number} days?
4	On days when I slept for more than {number} minutes, what was my average running speed?
5	What was my highest heart rate variability over the past {number} days?
6	What was the highest number of steps I took in the last number days?
7	How many times did I exercise today?
8	What was my average number of deep sleep minutes?
9	What is the standard deviation of my percentage of deep sleep?
10	What was my average awake percentage over the past {number} days?
11	What was the standard deviation of my deep sleep minutes over the past {number} days?
12	What was the duration of my last run?
13	What was my median percentage of deep sleep over the past {number} days?
14	What is the total time I spent on the treadmill for workouts lasting less than {number} minutes?
15	How many days did I participate in aerobic workouts during the last {number} days?
16	What is the total number of steps I took during my workouts in the last {number} days?
17	On days when I have less than {number} minutes of deep sleep, what is my average distance on the treadmill?
18	How many days have I run in the past {number} days?
19	What was my lowest sleep duration over the past {number} days?
20	How many days have I slept for at least {number} minutes in the last {number} days?
21	What was the total number of calories I burned during my last {number} runs within the past {number} days?
22	On days when I slept for at least {number} minutes, what is my total number of steps taken during runs?
23	What was the median number of steps I took yesterday?
24	What is the standard deviation of my deep sleep percentage over the past {number} days?
25	What was my average heart rate during my last aerobic workout?
26	What was my highest number of deep sleep minutes?
27	How many days did I sleep for less than {number} minutes?
28	How many times have I exercised in the last {number} days?
29	What was the duration of my longest run within the last {number} days?
30	What has been my average percentage of light sleep over the past {number} days?

Table A.1: Sample Objective Personal Health Queries. A selection of objective personal health queries that were generated as described in subsection 3.2.1.

A.5 Open-Ended Personal Health Insights Queries

A.5.1 Sample Queries

#	Query
1	How does my Stress Score correlate with my daily Steps?
2	How am I tracking towards my long term goals, as it relates to improving stress/sleep?
3	What are my personal bests for running speed, distance, and time?
4	How am I progressing in my fitness?
5	How has my meditation practice improved over time?
6	What are the differences in my sleep patterns on weekdays versus weekends?
7	What is the best amount for me to run?
8	How do I reduce stress?
9	What time of day do I feel most energized?
10	Are there specific days of the week when I tend to be more active or less active, and have these patterns remained consistent?
11	How does sleep duration affect heart rate recovery?
12	How is my deep sleep trending?
13	What is the relationship between my stress levels and my sleep quality?
14	Based on my age, what are the best exercises for me to do?
15	How is my bed time affected by steps last month?
16	Is the intensity of my workout related to my sleep quality?
17	How do I lose weight?
18	How does my sleep compare to the recommended sleep duration for someone in my age group?
19	Should I incorporate more cardio with my current physique?
20	Ideally how many days a week should I be working out?
21	Does doing abs everyday help my health goal?
22	What are some other factors that could be affecting my mental health condition, but that are not being tracked by my Fitbit?
23	How can I improve my muscle recovery?
24	What are the best exercises for runners?
25	What is my ideal fat burning workout?
26	What is my average calories burned per week?
27	What was my average bed time last month?
28	Is there a correlation between workout intensity and duration and the amount of deep sleep and REM sleep I am getting?
29	What was my lowest sleep duration last month?
30	What was my worst steps last month?

Table A.2: Sample Open-Ended Personal Health Queries. A selection of open-ended personal health queries that were generated as described in subsection 3.2.2.


A.5.2 Raw Data Translation

Here we present examples of raw data that were translated for human annotators evaluating the reasoning quality of our code generation baseline versus PHIA (subsection 3.4.2). In order to avoid encumbering human annotators without domain expertise (e.g., Pandas dataframes), raw data translation was necessary. This is in contrast to the code quality evaluation, in which the human annotators had adequate domain expertise and were able to evaluate data without translation.

Question:

What are anomalies in my breathing rate last month?

Code Generation Response:

Act: Python 

```

last_month = (pd.Timestamp.now() - pd.DateOffset(months=1))
last_month_data = summary_df[summary_df.index > last_month]
# Calculate the mean and standard deviation of breathing rate
mean_breathing_rate = last_month_data['breathing_rate'].mean()
std_breathing_rate = last_month_data['breathing_rate'].std()
# Identify anomalies as values outside of 2 standard deviations from the mean
anomalies = last_month_data[abs(last_month_data['breathing_rate'] - mean_breathing_rate) > 2 * std_breathing_rate]
anomalies

```

Observe:

ERROR: KeyError: 'breathing_rate'

Answer:

This question cannot be answered from the provided dataset as it does not contain breathing rate data.

Translation:

Overall Goal: This code aims to identify anomalies in the user's breathing rate data for the past month.

Basic Logic: The code calculates the average and standard deviation of the breathing rate data. It then identifies data points that are more than two standard deviations away from the average as anomalies.

Final Deliverable: The code attempts to provide a list of anomalous breathing rate data points. However, in this case, it encounters an error because the dataset does not contain breathing rate data, and the code cannot complete the task.

Figure A.8: In this example PHIA generates code to look for anomalies in the user's breathing rate in the last month. The translation captures PHIA's intention and the attempt which failed after code generation because breathing rate data is not a part of the available data.

Question:

How does my stress impact my workout performance?

Code Generation Response:**Act: Python**

```

activities_df.index = activities_df.index.date
summary_df['date'] = summary_df.index.date
# Shifting stress management score data forward by one day
summary_df["next_days_stress_management_score"] = summary_df["stress_management_score"].shift(-1)
# Joining the dataframes
joined_df = summary_df.join(activities_df, how='right', lsuffix='_activity')
# Calculating the correlation between stress management score and calories burned
corr_stress_calories = joined_df['stress_management_score'].corr(joined_df['calories'])
# Calculating the correlation between stress management score and active zone minutes
corr_stress_active_minutes = joined_df['stress_management_score'].corr(joined_df['active_zone_minutes'])
(corr_stress_calories, corr_stress_active_minutes)

```

Observe:

(0.469, 0.099)

Answer:

The correlation between your stress management score and calories burned is 0.47, indicating a moderate positive relationship. This suggests that as your stress levels decrease, you tend to burn more calories during workouts. However, the correlation between stress management score and active zone minutes is only 0.10, indicating a weak relationship. This suggests that your stress levels may not have a significant impact on the intensity of your workouts.

Translation:

Overall Goal: This code aims to determine the relationship between stress levels and workout performance.

Basic Logic: The code first combines data on stress levels and workout activity. Then, it calculates the correlation between stress levels and two workout metrics: calories burned and active zone minutes. A correlation measures how closely two variables are related.

Final Deliverable: The code outputs the correlation values between stress levels and each workout metric. A positive correlation suggests that as stress levels decrease, workout performance improves (for example, more calories burned). A weak correlation indicates that stress levels may not significantly impact workout performance.

Figure A.9: In this example PHIA generates code to analyze the relationship between the user's stress, as indicated by the stress management score, and the user's workout performance, as indicated by calories burned and active zone minutes. The translation captures PHIA's intention to determine the relationship between stress levels and workout performance with the data available to it, as well as PHIA's success in generating a reasonable answer based on the computed correlations.

A.5.3 Annotator Rubrics

Table A.3 and Table A.4 detail the rubrics given to annotators in order to evaluate the performance of both the code generation baseline and PHIA in terms of reasoning quality and code quality (subsection 3.4.2) respectively. Table A.3 focuses on evaluating the relevance,

domain knowledge, harmfulness (reformulated as 'Avoids Harm' in our results in Chapter A, Section 3.4), and clarity of textual responses. In contrast, Table A.4 focuses on evaluating the ability to avoid hallucinations and appropriate usage of columns. Both rubrics also evaluate for quality with respect to interpretation and personalization, as well as an overall measure of quality.

Dimension	Question and Options	Comments
Overall Reasoning	What is the overall quality of this response? Options: 1-5	<ol style="list-style-type: none"> 1. Very Bad: The model did not properly comprehend or answer the users question in any way. 2. Poor: The model may have partially understood the question but did not respond very well. 3. Acceptable: The model mostly understood the question but did not answer in a comprehensive manner. 4. Good: The model understood the question and provided all necessary information. 5. Excellent: The model understood the question and goes above and beyond in its logic and answer.
Relevance	Does this response reference all relevant user data? Options: Yes, No	Relevant data is any data that is interesting or useful for developing the recommendations (sleep) or summary (readiness)
Interpretation	Does this response properly interpret referenced user data? Options: Yes, No	Accounts for proper interpretations and missing interpretations.
Personalization	Does this response contains evidence of correct personalization, reference appropriate user data, or correctly refuse to answer when such data is missing? Options: Yes, No	Accounts for the use of user data in order to tailor insights to that user instead of just using generic ones.
Domain Knowledge	Does this response contain evidence of relevant and correct domain knowledge (e.g., mention of a relevant and/or correct fact for answering the question)? Options: Yes, No	
Logic	Does this response contain evidence of correct reasoning and logic? Options: Yes, No	
Harmfulness	Does this response contains information that could lead to harm? Options: Yes, No	
Clarity	Is this response easy to read and is it clear about the actions the user can perform? Options: Yes, No	

Table A.3: Reasoning Quality Rubric. Questions used for annotating the reasoning quality (subsection 3.4.2) of final answers.

Dimension	Question and Options	Comments
Overall Quality of Code	What is the overall quality of the code in this response? Options: 1-5	<ol style="list-style-type: none"> 1. Very Bad: The model did not properly comprehend or answer the users question in any way. 2. Poor: The model may have partially understood the question but did not respond very well. 3. Acceptable: The model mostly understood the question but did not answer in a comprehensive manner. 4. Good: The model understood the question and provided all necessary information. 5. Excellent: The model understood the question and goes above and beyond in its logic and answer.
Avoids Hallucination	Does the final answer avoid hallucination? Options: Yes, No, N/A	In some cases the language model will hallucinate data. For example, it might compute an average sleep duration of 300 minutes and call this 8.3 hours instead of 6. Or, it might reference data that it doesn't have access to, like the user's BMI
Column Usage	Does the agent use the correct columns? Options: Yes, No, N/A	You might reply "No" to this question if the model used the <code>heart_rate_variability</code> column to answer a question about "average heart rate".
Time Usage	Does the agent use the correct time frame? Options: Yes, No, N/A	For example, if the user asks "what is my average step count over the last 30 days" and the agent uses code that computes the average over the entire duration it has data this would be a "No".
Interpretation	Does the agent's code correctly interpret the question? Options: Yes, No, N/A	Regardless of whether or not the agent's code executed without bugs, did the generated code accurately attempt to address the question?
Personalization	Does the final answer show evidence of personalization? Options: Yes, No, N/A	The bar for personalization is high. We define it as "a decision or recommendation that may not be generated for a user with different data". For example, if the question is "Do I run enough" and the answer is "you ran three times this week" we would answer "No". On the other hand, if the answer was "You run three times a week and that's a healthy amount" the answer would be "Yes".

Table A.4: Code Quality Rubric. Questions used for annotating the code quality (subsection 3.4.2) of final answers.

A.5.4 Inter-Rater Agreement

In order to gauge the reliability of the ratings provided, we used Bennett’s S-Score [Bennett et al. \[1954\]](#) which is especially useful to assess how consistent individuals are in making categorical judgments. Bennett’s S-Score takes into account the number of categories into which responses are being classified and the distribution of ratings across these categories. Bennett’s S-Score is in a range of -1 to 1, with a score below 0 indicating worse than random chance, a score of 0 indicating random chance, and a score above 0 indicating better than random chance. For example, a score of 0.683 shows that the agreement among the raters is 68.3% better than what would be expected by random chance alone and is a considerably good degree of agreement. Table A.5 summarizes the inter-rater agreement using Bennett’s S-Score for seven categories of human ratings on query responses and five categories of human ratings on code generations.

Question	Bennett’s S-Score
Reasoning	
Relevance	.538
Interpretation	.683
Personalization	.654
Domain Knowledge	.208
Logic	.718
Harmfulness	.972
Clarity	.505
Code Quality	
Avoids Hallucination	.529
Column Usage	.622
Time Usage	.520
Interpretation	.617
Personalization	.348

Table A.5: Inter-Rater Agreement. Bennett’s S Scores for human ratings of the query responses and code generations.

A.6 Synthetic Wearable Users

A.6.1 Data Schema

Table A.6 and Table A.7 correspond to descriptions of daily summary data and activities data respectively. This is structured data that both our baselines and PHIA view and process as a part of their workflow. In Table A.6, each row corresponds to a single day's data for an individual user, encompassing a range of indicators from basic steps taken to detailed sleep analysis and heart rate metrics. Table A.7 contains detailed metrics for each activity session, including start and end times, the type of activity (e.g., running, biking, weightlifting), and performance statistics such as distance covered, elevation gain, and calories burned.

Column Name	Datatype	Description
<code>datetime</code>	date	The day the data describes
<code>steps</code>	integer	The number of steps taken during the day
<code>sleep_minutes</code>	integer	The total number of minutes of sleep from the night before.
<code>bed_time</code>	timestamp	The time the user went to sleep the night before.
<code>wake_up_time</code>	timestamp	The time the user woke up that morning.
<code>resting_heart_rate</code>	integer	The measured resting heart rate for that day.
<code>heart_rate_variability</code>	float	Heart rate variability, measured in milliseconds, for that day.
<code>active_zone_minutes</code>	integer	The number of active zone minutes (minutes with elevated heart rate) for that day.
<code>deep_sleep_minutes</code>	integer	The total number of minutes spent in deep sleep the night before.
<code>rem_sleep_minutes</code>	integer	The total number of minutes of REM sleep from the night before.
<code>light_sleep_minutes</code>	integer	The total number of minutes spent in light sleep the night before.
<code>awake_minutes</code>	integer	The total of minutes spent awake during last night's sleep period.
<code>deep_sleep_percent</code>	float	The fraction of last night's sleep period spent in deep sleep.
<code>rem_sleep_percent</code>	float	The fraction of last night's sleep period spent in REM sleep.
<code>light_sleep_percent</code>	float	The fraction of last night's sleep period spent in light sleep.
<code>awake_percent</code>	float	The fraction of last night's sleep period spent awake.
<code>light_sleep_percent</code>	float	The fraction of last night's sleep period spent in light sleep.
<code>stress_management_score</code>	integer	The stress management score measures how the user responds to stress based on their heart rate, sleep, and activity level data. A higher score is "better".
<code>fatburn_active_zone_minutes</code>	integer	The number of active zone minutes spent in the "fatburn" heart rate zone.
<code>cardio_active_zone_minutes</code>	integer	The number of active zone minutes spent in the "cardio" heart rate zone.
<code>peak_active_zone_minutes</code>	integer	The total number of minutes spent in the "peak" - or highest activity - zone.

Table A.6: Daily Summary Table Schema. Columns, data types and data descriptions in the Daily Summary table.

Column Name	Datatype	Description
<code>startTime</code>	timestamp	The timestamp of the start of the activity.
<code>endTime</code>	timestamp	The timestamp of the end of the activity.
<code>activityName</code>	string	The type of activity. This is one of ['Outdoor Bike', 'Run', 'Bike', 'Aerobic Workout', 'Weights', 'Elliptical', 'Yoga', 'Spinning', 'Treadmill'].
<code>distance</code>	integer	The distance (in meters) covered by the user during the activity.
<code>duration</code>	integer	The duration of the activity in minutes.
<code>elevationGain</code>	integer	The number of meters of elevation gain during this activity.
<code>averageHeartRate</code>	integer	The average heart rate during this activity.
<code>calories</code>	integer	The number of calories burned during this activity.
<code>steps</code>	integer	The total number of steps taken during this activity.
<code>activeZoneMinutes</code>	int	The total number of active zone (higher heart rate) minutes during this activity.
<code>speed</code>	float	The average speed (in m/s) during this activity.

Table A.7: Activities Table Schema. Columns, data types and data descriptions in the Activities table.