
Subject: The paper I sent earlier
Date: Wednesday, July 2, 2025 at 12:47:17 PM Central Daylight Time
From: Martin Paulus
To: Dale Peasley, Mikey Ferguson

Hi guys
I have been looking at the paper and had my GPT friend put a more concrete set of steps together how we might use this with our data:

Wednesday, July 2, 2025: Update based on the "*Large Language Models are Powerful Electronic Health Record Encoders*" paper

Below is a scientist-oriented snapshot of the study's methods and quantitative findings.

- **Purpose** – Test whether general-purpose, instruction-tuned LLM embedding models can encode structured EHRs for clinical prediction, versus an EHR-specific foundation model (CLMBR-T-Base) and a counts + GBM baseline.
- **Datasets**
  - *EHRSHOT* (Stanford): 6,739 patients | 921 k visits | 41.7 M events
  - *UK Biobank* (external): 387,464 patients | 19.5 M visits | 72.3 M events
- **Benchmark tasks** – 15 prediction tasks in four groups (operational outcomes, lab abnormalities, new diagnoses, chest-X-ray findings), evaluated in few-shot regimes (k = 1–128 per class).
- **EHR-to-text serialization** – Structured records converted to Markdown (≤ 4,096 tokens): demographics → 24 key LOINC vitals/labs → visit summary → reverse-chronological event lists with ontology-based code descriptions.
- **Embedding models & training**
  - LLM encoders: **GTE-Qwen2-7B-Instruct** & **LLM2Vec-Llama-3.1-8B-Instruct** generate patient embeddings; logistic-regression heads used for all tasks.
  - Baselines: CLMBR-T-Base embeddings and counts + GBM evaluated in the same pipeline.
- **EHRSHOT performance (macro-AUROC, 95 % CI)**

Model	**Macro-AUROC**	**With CLMBR concat**
GTE-Qwen2-7B	0.755 (0.730–0.780)	**0.786 (0.762–0.811)**
LLM2Vec-Llama-3.1-8B	0.734 (0.707–0.762)	**0.773 (0.747–0.799)**
CLMBR-T-Base	0.769 (0.746–0.792)	—
Counts + GBM	0.719 (0.691–0.748)	—

- **UK Biobank external validation (macro-AUROC)**

Task group	GTE-Qwen2-7B	CLMBR-T-Base	Llama-8B
Mortality	**0.826**	0.782	0.781
Hospitalization	**0.655**	0.639	0.638
23-disease onset	**0.727**	0.707	0.715
Macro avg.	**0.736**	0.709	0.711

- **Few-shot behavior** – LLM embeddings yield the largest gains over CLMBR and GBM when 4–32 labeled examples are available; benefits taper as data scale increases.
- **Context & recency effects**

• Best macro-AUROC for GTE at 4,096 tokens; Llama peaks at 2,048 tokens, with no gain from 8,192 tokens.
• Limiting input to the most recent **1 month** of history performs on par with full histories, hinting at stronger predictive signal in recent events.

- **Limitations flagged by authors** – Hand-crafted serialization may bias comparisons; 4 k-token cap can omit long-term context; GPU cost high (≈ 20 h on 8 × A100 for GTE; ≈ 30 h for Llama); results drawn from only two health-system datasets.

Step-by-step blueprint for turning structured EHRs into GPT-ready inputs and predictive features

1. **Anchor the clinical question up front**
   Pick a reference ("prediction-time") date and define the outcome window—e.g., one-year mortality or 30-day readmission. Normalizing every event timestamp to that single date simplifies temporality downstream.
2. **Extract & harmonize raw codes**
   Pull all structured tables (diagnoses, meds, labs, procedures, vitals) and map every code to a common "ontology/code" string. Resolve code → description via prepared ontologies (SNOMED, RxNorm, LOINC, CPT-4, ICD-10-PCS, etc.) and add custom maps for gaps.
3. **Serialize the record into Markdown**
   Use a lightweight, strictly ordered text template because Markdown "adds minimal overhead" yet preserves structure for the model.
   1. # Electronic Healthcare Record
   2. Current time: [YYYY-MM-DD]
   3. ## Patient Demographics
   4. …
4. **Populate the template (information-rich → general)**
   4.1 **Demographics first** (age, sex, ethnicity). Convert DOB to age.
   4.2 **Recent high-value time series**: choose ~24 canonical vitals/labs, keep the last three values, add units plus low/normal/high flags.
   4.3 **Visit summaries**: one-line bullets of each visit to guard against later truncation.
   4.4 **Detailed reverse-chronological events** (conditions ▸ meds ▸ procedures).
5. **Cap length & control recency**
   Limit the serialized text to the context window you can afford (e.g., 4 096 tokens). If GPU memory allows, test longer windows (8 192) or chunk averaging; in practice, a **one-month** window often equals full-history performance.
6. **(Optional) Prepend task-specific instructions**
   Short natural-language prompts ("Predict whether the patient will be hospitalized in the next year") can boost embeddings—verify via ablation by removing or generalizing instructions.
7. **Choose an LLM embedding backbone**
   Select a decoder-only model already converted to an embedding variant (e.g., **GTE-Qwen2-7B-Instruct** or **LLM2Vec-Llama-3.1-8B**) that can ingest ≥4 k tokens.
   Feed the serialized text; mean-pool the last hidden layer to obtain a fixed-length vector.
8. **Attach a lightweight predictor**
   Logistic regression or gradient-boosted trees on top of the frozen embeddings give calibrated probabilities while keeping training cheap.
9. **Evaluate systematically**
   Follow a few-shot protocol (k = 1→128 positives + negatives) and report AUROC/AUPRC with

bootstrapped 95 % CIs.

10. **Run ablations & sensitivity analyses**
    Remove demographics, aggregated labs, visit summaries, or instructions one at a time to see which components drive performance.
11. **Plan for production constraints**
    Expect ~20 h on 8×A100 GPUs for 6 k patients at 4 k tokens with a 7-B model; budget >30 h for an 8-B model or a 400 k-patient external cohort.
12. **Document limitations & future refinements**
    Manual serialization can bias results; token limits may drop older context; large models add latency and cost. Consider serialization-free approaches or model distillation for deployment.

In practice: start with Steps 1–5 to create a reproducible, compact Markdown view of each patient; plug that into Step 7's embedding model; and iterate through Steps 8–12 until the predictive metrics and runtime satisfy your clinical use-case.

Critical implementation issues highlighted by the manuscript

Domain	***What to watch for***	***Evidence from the paper***
1. Data preprocessing & serialization	• *Subjective template design*: manually deciding which tables, time windows, and concepts go first can bias the model toward the target tasks. • *Ontology mapping headaches*: every OMOP/SNOMED/ICD/RxNorm/LOINC code must be mapped to a human-readable string—rare or site-specific codes easily slip through the cracks. • *Signal-to-noise filtering*: the authors had to merge synonyms, pick 24 "canonical" vitals/labs, and clip implausible lab values. Skipping a similar quality-control pass will flood the tokenizer with junk tokens.	
2. Context length & temporal coverage	• A hard 4 096-token cap means older visits may be truncated. • Ablations show that **one-month of history often matches or beats the full record**—so blindly serializing the entire chart wastes tokens and GPU.	
3. Instruction engineering	• Task-specific prompts materially change embedding quality; they also inflate the workload because each (patient × task) pair must be re-encoded. • Performance is sensitive to the *exact wording* of those prompts, which hurts reproducibility across sites or versions.	
4. Model architecture & downstream heads	• LLM embeddings **require a separate classifier** (logistic regression in the study); you lose the native zero-shot capacity that makes GPT attractive. • Embeddings and CLMBR vectors are complementary—concatenating them gave the single best AUROC. Plan for model ensembling if you want peak accuracy.	
5. Compute &	• Encoding 6 739 patients (EHRSHOT) at 4 096 tokens	

latency costs	needed **≈ 20 h on 8 × A100 GPUs** for a 7-B model; scaling to 387 k UK Biobank patients took **≈ 35 h per task**. • Larger models (8 B) hit memory errors unless you batch-split, further slowing the pipeline.
6. Generalization limits	• The study tested only two datasets; differences in coding practices or data density at other hospitals may erode gains. • CLMBR’s drop on UK Biobank underscores how *unseen codes* hurt specialized models—LLM approaches mitigate but don’t eliminate this risk.
7. Bias, fairness & calibration	• Manual choices about what to include first (e.g., labs vs. medications) can privilege some conditions over others. • LLMs are agnostic to coding systems, but their embeddings still need calibration layers; otherwise probability outputs are hard to trust clinically.

Practical take-aways

1. **Design serialization before training**: lock the template, code maps, and time window, then version-control it to curb “template drift.”
2. **Pilot recency windows**: start with 30 days and expand only if specific tasks demand long-term context.
3. **Budget GPUs realistically**: a single full-scale encode pass can be a multi-day cluster job—plan queuing and fault-tolerance.
4. **Keep a calibration layer**: temperature scale or isotonic-regress the logistic head so bedside probabilities remain reliable.
5. **Document every prompt**: treat instruction texts as hyper-parameters; store them alongside model weights for auditability.

Addressing these points early will spare costly re-runs and smooth the path from proof-of-concept to a deployable clinical prediction service.

Below is a concrete, end-to-end blueprint for turning structured EHR data into a single GPT-friendly text string that can be embedded and used to predict **12-month depression deterioration** (e.g., PHQ-9 ≥ 15 or new MDD hospitalization). It is written as if you were about to implement it tomorrow on a real data warehouse; feel free to copy-paste the code blocks as starting templates.

---

1 Fix the prediction target and index time

```
index_date   = visit_date         # index = most recent outpatient encounter
outcome_horizon = 365            # days to look forward
label = 1 if phq9>=15 or mdd_hosp within 365d else 0
```

Rationale – anchoring every record to a single date lets you standardize “past-to-future” directionality and build rolling cohorts easily.

2 Pull & harmonize raw tables (OMOP style)

Domain	***Key fields kept***	***Extras***
Condition	condition_concept_id, start_date	map ICD-10 F32.*, F33.* → “MDD”
Drug exposure	drug_concept_id, start_date, days_supply	SSRIs, SNRIs, antipsychotics
Measurement	loinc_concept_id, value, unit, date	TSH, HbA1c, CRP, vit D, LDL
Observation	observation_concept_id, value	PHQ-9, GAD-7, BMI, smoking status

Visit	visit_concept_id, date, type	“Psychiatry-OP”, “ED”, “Inpatient”

Convert concept_ids to **short human-readable tokens** (one time only).
Example:

```
SELECT concept_id, LOWER(REPLACE(concept_name,' ','_')) AS token
INTO concept_tokens
FROM vocabulary.concept
```

3 Shallow feature engineering

1. **Temporal bucketing** – tag each event with Δt = index_date – event_date in days.
2. **Value flagging** – for every continuous measure create discrete flags: low/normal/high.
3. **Medication adherence surrogate** – rolling 90-day medication possession ratio (MPR) for antidepressants.

4 Serialize into deterministic Markdown (≤ 4 096 tokens)

```
# ELECTRONIC_HEALTH_RECORD
current_time: 2025-03-14
prediction_window: next_365_days
## PATIENT
age: 46
sex: female
race: white
## RECENT_METRICS
bmi: 31 (high)
phq9: 12 (moderate)
gad7: 7 (mild)
tsh: 3.8 µIU/mL (normal)
## MED_HISTORY_LAST_365_DAYS
[−10d] condition: major_depressive_disorder (F33.1)
[−10d] drug_start: sertraline 50 mg qd (days_supply=30)
[−45d] measurement: crp 6 mg/L (high)
[−60d] visit: psychiatry_outpatient
[−120d] measurement: hba1c 7.2 % (high)
[−200d] drug_start: bupropion_sr 150 mg bid
## MED_HISTORY_−366_to_−1095_DAYS
… (truncate middle events if token budget exceeded)
## SOCIAL
smoking: former
alcohol: occasional
exercise_level: low
```

Determinism is crucial – the same patient on the same date always yields identical text.

5 Embed once per patient

```
from transformers import AutoTokenizer, AutoModel
tok   = AutoTokenizer.from_pretrained("thenlper/gte-small")   # 8 k ctx
model = AutoModel.from_pretrained("thenlper/gte-small")
tokens = tok(ehr_markdown, truncation=True, padding=False, return_tensors='pt')
emb    = model(**tokens).last_hidden_state.mean(dim=1)       # shape (1, 768)
```

Tip – cache the embedding table; downstream experiments become a logistic-regression fit, not a

GPU re-encode.

6 Lightweight predictor

```
from sklearn.linear_model import LogisticRegressionCV
clf = LogisticRegressionCV(Cs=10, cv=5, max_iter=5000, class_weight='balanced')
clf.fit(embeddings_train, y_train)
y_hat = clf.predict_proba(embeddings_test)[:,1]
```

Evaluate AUROC/AUPRC and calibration (e.g., Hosmer–Lemeshow, ECE).

Keep a temperature-scaling layer if probabilities will drive clinical alerts.

7 Ablation tests (one line each)

```
python run.py --drop-section RECENT_METRICS
python run.py --window 90   # only 3 mo history
python run.py --remove-labs
python run.py --strip-meds
```

Track which chunks move performance the most; typical finding: **recent PHQ-9 trajectory + antidepressant MPR** dominates.

8 Deployment checklist

Item	**Why it matters**	**Quick mitigation**
Token cap	4096-token GPT-4o window ≈ 25 k chars	enforce size guard, drop oldest events
PHI risk	Markdown is still PHI	encrypt at rest, purge names/addresses
Latency	150 ms per patient (7-B on A100, fp16)	nightly batch encode; serve cached vector
Drift	Coding practice changes (ICD-11, new meds)	weekly concept-map diff + re-train trigger

TL;DR

1. **Anchor** each chart at the prediction visit.
2. **Map** all codes to short tokens.
3. **Serialize** in a fixed Markdown scaffold (NEWEST → OLDEST).
4. **Embed** with a 7-B instruction-tuned model once.
5. **Fit** a simple classifier; iterate with ablations.

With one evening of SQL + Python you'll have a solid, auditable depression-risk encoder that can be swapped into any GPT-friendly pipeline.

Turning the GPT-based EHR encoder into a digital-twin factory

Below is a concrete technical pathway that starts with the Markdown-serialization/LLM-embedding pipeline we just designed for 12-month depression risk and extends it into **individualized, counter-factual simulators**—i.e., digital twins that let you ask *"what if I adjust medication X, start CBT today, or change BMI by 3 kg/m*^{2}*—how does that alter this patient's future PHQ-9 or hospitalization risk?"*

Stage	***Objective***	***Design choices & tips***
1. Canonical state vector	Freeze the **baseline encoder** you already validated (Steps 1–5 earlier). Each patient-timepoint is a 768-D vector **st** capturing *current*	• Embed *every* monthly snapshot for the past N months (or per visit).

	demographics, labs, meds, diagnoses, recent PHQ-9 trajectory, etc.	• Store **(patient_id, t, st)** in a feature store so you can query history fast.
2. Action vocabulary	Define a compact, machine-readable set of *interventions* **at**: SSRIs↑/↓, start CBT, add bupropion, ↑exercise flag, BMI–3 %, etc.	• Keep ≤ 50 discrete actions to avoid data sparsity. • Encode "no change" as an explicit action.
3. Dynamics model (st, at) → st+Δ	Learn how state evolves under actions. Two viable architectures: **(a) GPT-style autoregressive transformer** that inputs *serialized Markdown +* tokens and predicts next-month Markdown, then re-encode. **(b) Lightweight MLP/RNN** that takes *(st ⊕ at)* and outputs Δs.Pick (b) if speed matters; (a) if richness of generated notes is critical.	• Train with teacher forcing on real sequences: minimize MSE for continuous labs, CE for discrete codes, KL for embedding drift. • Include a *calendar offset* embedding so the model knows Δ days.
4. Outcome heads	Attach differentiable heads to **s** that predict clinical endpoints (PHQ-9, inpatient MDD stay, suicidality flag). You already have a logistic-regression head; keep it frozen or co-train.	• Multi-task learning stabilizes the dynamics model—every future state is supervised by all available outcomes.
5. Twin generator	For a real patient at t = 0: 1. Encode to **s0**. 2. Choose counter-factual action sequence **{â0:T-1}** (e.g., "increase SSRI dose at t = 0, add CBT at t = 1"). 3. Roll the dynamics model forward T steps → **ŝ1:T**. 4. Pipe each **ŝt** through the outcome heads to yield risk trajectories.	• Sampling: add Gaussian noise to latent Δs or use nucleus sampling on the autoregressive model to capture uncertainty. • Generate N = 100 twins per strategy and average risks to estimate credible intervals.
6. Calibration & realism checks	**Internal validity**: compare simulated trajectories with held-out real patients who actually received those actions.**Face validity**: have clinicians judge generated Markdown snippets (if using the autoregressive path).	• Use *energy distance* or *K-nearest-neighbor MMD* between real and synthetic sequences. • Track coverage of lab trajectories (e.g., HbA1c cannot jump 5 % in a month).
7. Policy evaluation loop	Wrap the twin generator in an API: get_twin(patient_id, action_plan_json, horizon=365) → JSON of monthly outcome probabilities & synthetic notes.	• Cache embeddings; only the dynamics roll-out needs GPU (often feasible on a single A10). • Add a Shapley or Integrated-Gradients explainer on **ŝt** to surface *why* risk changed.
8.	Digital twins drive decision support, so document assumptions & biases: coding drift,	• Maintain model cards with: data window, action

Governance & safety	selection bias (only coded depression), token-limit truncation.	coverage, out-of-distribution detector (e.g., Mahalanobis on **s**).

Minimal working code sketch (MLP dynamics)

```
# s_dim = 768, a_dim = 50 one-hot, hidden = 1024
dyn = torch.nn.Sequential(
  torch.nn.Linear(768 + 50, 1024), torch.nn.ReLU(),
  torch.nn.Linear(1024, 768)
)

for (s_t, a_t, s_next) in dataloader:
  pred = s_t + dyn(torch.cat([s_t, a_t], dim=-1))   # residual update
  loss = torch.nn.MSELoss()(pred, s_next)
  loss.backward(); opt.step()
```

Generate a twin:

```
s = s_0.clone()
traj, risks = [], []
for a in planned_actions:              # list of one-hot vectors
  s = s + dyn(torch.cat([s, a], -1))
  traj.append(s)
  risks.append(outcome_head(s).sigmoid())
```

Why this qualifies as a digital twin

1. **Individualized** – seeded with the patient's own encoded EHR state.
2. **Counter-factual** – you can swap in any plausible action sequence.
3. **Dynamic** – generates month-by-month physiological and diagnostic evolution, not just a static risk score.
4. **Interpretable** – outcome heads plus attribution on latent dimensions show which features drive divergence.

With this pipeline, a clinician can interactively explore *"What if we augment sertraline with bupropion?"* and immediately see projected PHQ-9 trajectories, hospitalization risk, and even synthetic progress-note snippets—all while the underlying state vectors remain de-identified and compact enough for real-time use.

The Art of War: In the midst of chaos, there is also opportunity.
Martin

Scientific Director and President, Laureate Institute for Brain Research
6655 S Yale Ave Tulsa, OK 74136-3326 P 918 502 5120 F 918 502 5135
email: mpaulus@laureateinstitute.org X:@mpwpaulus
web: http://www.laureateinstitute.org

Professor of Neuroscience, Oxley College of Health and Natural Sciences
University of Tulsa, email: mpp4692@utulsa.edu

Adjunct Professor, Department of Psychiatry, University of California, San Diego
email: mpaulus@health.ucsd.edu