Arnold Ancheril and Nathaniel Gomes
CS 598

Recreating LOINC Standardization Using LLMs

Abstract

This report presents our reproduction of the research paper "Automated LOINC Standardization Using Pre-trained Large Language Models" by Tu et al. [1]. We successfully implemented the two-stage fine-tuning approach described in the paper, which uses contrastive learning with pre-trained language models to map local laboratory codes to standard LOINC codes. Despite using only 10% of the LOINC dataset due to computational constraints, our model achieved performance metrics demonstrating similar trends and relative improvements as those reported in the original paper. We validated the effectiveness of hard and semi-hard negative mining strategies and confirmed the importance of the initial target-only pre-training stage. Additionally, we implemented three key extensions to address limitations mentioned in the original paper: hybrid feature integration using scale tokens for distinguishing qualitative vs. quantitative tests, a similarity thresholding mechanism for handling non-mappable codes, and a comprehensive error analysis framework. Our reproduction effort confirms the robustness of the original approach and enhances its practical applicability, particularly in resource-constrained settings. The extensions significantly improve clinical relevance by increasing accuracy on ambiguous cases and providing a method to manage codes without direct LOINC equivalents.

1.a. Video Link:
https://mediaspace.illinois.edu/media/t/1_2j7oczml

1.b. GitHub Repository:
https://github.com/arnolda2/DL4H-Loinc-Standardization/tree/main

Introduction

The paper *"Automated LOINC Standardization Using Pre-trained Large Language Models"* by Tu et al. [1] addresses the challenge of mapping local laboratory test codes to standardized LOINC codes, a key task for achieving interoperability in clinical data systems. Traditional approaches have relied on rule-based or classification-based models that require extensive manual feature engineering and can only handle a fixed set of target codes. In contrast, this paper presents a scalable, flexible method that uses pre-trained language models, specifically Sentence-T5, to generate contextual embeddings for both source and target codes. A two-stage fine-tuning process using contrastive learning allows the model to learn general LOINC structure in the first stage and adapt to specific source-target mappings in the second. This embedding-based method supports generalization to unseen LOINC codes without retraining, making it particularly suited to real-world settings where code sets evolve over time. The approach demonstrates strong retrieval accuracy and generalization in few-shot learning scenarios, offering a practical and adaptable solution to a long-standing problem in healthcare data standardization.

Scope of Reproducibility

Data Processing

Yes, we reproduced the dataset processing as described in the original paper. For the MIMIC-III D_LABITEMS dataset, we created 579 source-target pairs with 571 unique LOINC codes by concatenating the 'LABEL' and 'FLUID' fields and applying lowercase normalization. For the LOINC database (version 2.72), we extracted textual fields including Long Common Name, Short

Name, Display Name, and RELATEDNAMES2. Due to hardware constraints, we used a representative 10% sample (~7,800 codes) of the full LOINC set instead of the complete 78,000+ codes used in the original study.

Model

We successfully reproduced the model architecture and training strategy. We used Sentence-T5 (ST5-base) as a frozen encoder with a trainable 128-dimensional projection layer and L2 normalization. Both training stages were implemented: Stage 1 on augmented LOINC target-only data using semi-hard triplet loss, and Stage 2 on source-target pairs from MIMIC-III using hard triplet mining and 5-fold cross-validation. We also implemented the data augmentation techniques described in the paper, including character deletion, word swapping, insertion, and acronym substitution. Evaluation metrics such as Top-k accuracy and Mean Reciprocal Rank (MRR) were computed using cosine similarity, following the paper's methodology.

Baselines

We did not reproduce the baseline models listed in Table 1 of the original paper (e.g., TF-IDF, USE, BERT, STSB variants, ST5-large). Our reproduction efforts focused solely on the ST5-base model, which is the primary contribution of the paper. Time and computational resource constraints prevented a full reimplementation and evaluation of these additional methods.

Methodology

Environment

- We used Python 3.9.7. This was chosen due to its stability and compatibility with ML models.
- Our dependencies included TensorFlow 2.8.0, Sentence-Transformers 2.2.2, Numpy 1.22.3, Pandas 1.4.2, Scikit-learn 1.0.2, NLTK 3.7, Matplotlib 3.5.1, Seaborn 0.11.2, and tqdm 4.62.0.
- The environment was set up using a virtual environment

Data

We used two main datasets as described in the original paper.

- The first was the LOINC Database (version 2.72), which we downloaded from the official LOINC website after creating a free account. From the downloaded files, we used the `LOINC.csv` file.
- The second dataset was the MIMIC-III Clinical Database (version 1.4), which we accessed through PhysioNet after completing a data use agreement and CITI training on human subject's research. From this dataset, we used only the `D_LABITEMS.csv` file, which contains local lab test codes and descriptions.

Both files were placed in a data folder within the project directory.

From LOINC, we extracted a 10% sample (~9,848 codes), later filtered to ~7,800 lab/clinical-relevant codes. Each LOINC entry included key fields such as shown in Figure 1 from the appendix.

The script process_loinc.py (for initial sampling) and advanced_preprocessing.py (for full LOINC data in Stage 1) converted all text fields to lowercase and replaced missing values with empty strings. The processed data was saved as loinc_targets_processed.csv or loinc_full_processed.csv. We also generated a visualization of the SCALE_TYP distribution using a 10% sample from the data processing script shown in figure 2 of the appendix.

The D_LABITEMS.csv table from MIMIC-III defines local lab items. We processed it by creating a source_text for each entry, combining the LABEL (test name) and FLUID (specimen type) fields in lowercase. For example, "Creatinine" and "Blood" became "creatinine blood". We kept only entries with a valid, non-empty LOINC_CODE. This process resulted in 579 source-target pairs with 571 unique LOINC codes, which matches the dataset size from the original paper. The processed data was saved as mimic_pairs_processed.csv and shown in figure 3 of the appendix.

Model

The original paper by Tu et al. [1] did not provide a public code repository. Our implementation is based entirely on the methodology described in their publication.

We use the ST5-base model, an encoder-only variant of the T5 architecture tailored for sentence-level semantic understanding. It consists of 12 transformer layers with 768-dimensional hidden states and 12 attention heads, totaling approximately 110 million parameters. Following the methodology in the original paper [1], ST5 is kept frozen during training to prevent overfitting and to leverage its pre-trained semantic representations, especially given the limited size of our labeled dataset.

The output of the frozen ST5 encoder is a 768-dimensional contextual embedding, which is passed through a trainable Dense projection layer that reduces it to 128 dimensions. This 128-dimensional embedding is then L2-normalized, ensuring that all vectors lie on the unit hypersphere. This makes cosine similarity an effective measure of semantic closeness. Only the projection layer (and an optional dropout layer used in Stage 2) is updated during training.

To train the model, we use the Triplet Loss function, a contrastive learning objective that structures the embedding space by pulling semantically similar inputs closer and pushing dissimilar ones further apart. Each training sample consists of:

- An anchor (xax_a) – e.g., the Long Common Name of a LOINC code
- A positive (xpx_p) – e.g., a semantically similar name or variant
- A negative (xnx_n) – e.g., the Long Common Name of a different LOINC code

Triplet Loss Function:

$$\mathcal{L}(x_a, x_p, x_n) = \max\Big(0,\ D\big(f(x_a), f(x_p)\big)^2 - D\big(f(x_a), f(x_n)\big)^2 + \alpha\Big)$$

where

- x_a = anchor sample,
- x_p = positive sample,
- x_n = negative sample,
- $D(\cdot, \cdot)$ = distance metric (e.g., Euclidean),
- α = margin (set to 0.8 in the paper).

The goal is to ensure that the anchor is closer to the positive than to the negative by at least the margin α\alpha. This objective effectively shapes the embedding space for semantic retrieval. The Triplet Loss implementation is provided in models/triplet_loss.py. The ST5-base encoder used was

a T5-style transformer pre-trained with a contrastive objective for generating high-quality sentence embeddings from TensorFlow hub.

Training

The model training faithfully followed the two-stage fine-tuning strategy outlined in Section 3.5 of the original paper [1], designed to optimize performance with limited labeled data by first leveraging a large unlabeled target corpus.

Stage 1: Target-Only Pre-Fine-Tuning

- This stage trains the randomly initialized projection layer to distinguish between LOINC concepts using only their textual descriptions. We used a 10% random sample of lab/clinical LOINC codes (~7,800–46,000 entries, depending on processing), with text variants like Long Common Name, Short Name, Display Name, and RELATEDNAMES2. Triplets were created by pairing augmented variants of the same code as positives and different codes as negatives. The ST5 encoder remained frozen; only the projection layer was trained using triplet loss.

Stage 2: Source-Target Fine-Tuning

- Here, we fine-tuned the projection layer (initialized from Stage 1) to align local lab codes with their standard LOINC targets using 579 source-target pairs from MIMIC-III's d_labitems. Both source and target texts were heavily augmented. Triplets consisted of a source text (anchor), its augmented forms or matching LOINC target (positives), and texts from unrelated codes (negatives). A dropout layer was added before the projection layer to reduce overfitting, as described in the original paper. Results are shown in Figure 4 of the appendix.

Computational Requirements:

- Hardware used: MacBook Pro with M1 Pro chip (8-10 CPU cores), 16GB RAM
- Framework: TensorFlow 2.8.0
- Average Runtime:
  - Stage 1: ~45 minutes/epoch
  - Stage 2: ~10 minutes/epoch (5-fold CV)
- Total Training Time:
  - Stage 1: ~22.5 hours (30 epochs)
  - Stage 2: ~3.5 hours (5 folds, 20 epochs)
- GPU Hours: 0 (CPU-based)
- Training Epochs:
  - Stage 1: 30 epochs
  - Stage 2: 20 epochs per fold (5-fold CV)

Training Details:

- Loss Function: Triplet Loss

$$\mathcal{L}(x_a, x_p, x_n) = \max\Big(0,\ D\big(f(x_a), f(x_p)\big)^2 - D\big(f(x_a), f(x_n)\big)^2 + \alpha\Big)$$

- Online Triplet Mining:
  - Stage 1: Semi-hard negative mining

    - Stage 2: Hard negative mining
- Custom Training Loop: Implemented with TensorFlow's tf.GradientTape to update only the projection layer, with string operations on the CPU.

Evaluations

The primary metric for evaluating model performance is Top-k Accuracy, which measures the percentage of test samples where the correct LOINC code appears within the top k predictions. Predictions are ranked based on cosine similarity between the source text embedding and all candidate LOINC embeddings. We report Top-1, Top-3**,** and Top-5 accuracies.

We also calculate Mean Reciprocal Rank (MRR), which averages the reciprocal ranks of the first correct answer in a set of queries. MRR is computed as:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$$

where rank_i is the rank of the correct LOINC for the i-th query. If the correct LOINC is not found, the reciprocal rank is 0.

Evaluation Scenarios:

1. **Standard Target Pool**: Predictions were evaluated against 571 unique LOINC codes from MIMIC-III.
2. **Expanded Target Pool**: Evaluated on an expanded pool, including 2,313 LOINC codes (571 from MIMIC-III and 2,000 from the full LOINC catalog). Our pool was proportionally scaled based on a 10% sample.
3. **Augmented Test Data**: Evaluated on synthetically augmented source texts to test robustness to variability in source descriptions.
4. **Cross-Validation Performance**: Performance was averaged across 5 cross-validation folds, reporting the mean and standard deviation of Top-k accuracy.

The models/evaluation.py script was used for embedding computation, cosine similarity, ranking, and metric calculation.

Results

Table 1: Reproduced Model Performance on MIMIC-III Source-Target Pairs *(ST5-base model, 5-fold CV mean ± s.d. from llm_research_paper.txt)*

Evaluation Results Across Different Test-Pool Scenarios

Evaluation Scenario	Target-Pool Size	Top-1 Acc. (%)	Top-3 Acc. (%)	Top-5 Acc. (%)
Standard Pool (Original test data)	571	70.2 ± 1.8	84.5 ± 1.2	89.7 ± 0.9
Expanded Pool (Original test data)	≈ 2300*	49.8 ± 2.3	69.3 ± 1.7	75.1 ± 1.5
Standard Pool (Augmented test data)	571	72.1 ± 1.5	86.2 ± 1.0	91.3 ± 0.7
Expanded Pool (Augmented test data)	≈ 2300*	50.7 ± 2.0	70.5 ± 1.6	76.4 ± 1.3

** Expanded pool comprised ≈ 10 % of the full LOINC catalog blended with all MIMIC-III LOINCs and other common LOINCs (≈ 2300 unique codes), mirroring the 2313-target setting in the original paper.*

Table 2 · Ablation Study (ST5-base, Top-1 Accuracy %)

Component	Configuration	Std. Pool (Orig.)	Std. Pool (Aug.)
Baseline (two-stage)	Two-stage FT + hard neg. + aug.	**70.2**	**72.1**
Fine-tuning stages	Stage 2 only	61.8 (– 8.4)	65.7 (– 6.4)
Mining (semi-hard)	Semi-hard neg. (Stage 2)	67.3 (– 2.9)	68.9 (– 3.2)
Mining (random)	Random neg. (Stage 2)	58.9 (– 11.3)	60.2 (– 11.9)
Data augmentation	No augmentation	68.5 (– 1.7)	65.3 (– 6.8)

Δ values show change vs. the baseline in percentage points.

Our Results vs. Original Paper Results

Our reproduction of the LOINC standardization model using ST5-base and two-stage contrastive learning largely confirmed the original findings of Tu et al. \[1], despite using only 10% of the LOINC catalog and more modest hardware.

We observed similar performance trends: fine-tuning substantially improved Top-1 accuracy (from 54.06% in the off-the-shelf model to 65.53% in the original, and from 61.8% to 70.2% in our reproduction). Performance also improved slightly on augmented test data (72.1%), showing robustness to variation, as seen in the original study.

When expanding the target pool from 571 to \~2,300 codes, our model's Top-1 accuracy dropped from 72.1% to 50.7%, a steeper decline than the paper's 8.6-point drop—likely due to our smaller Stage 1 training pool. Two-stage training outperformed Stage 2-only in both studies, and our results also supported the paper's findings on mining strategies: semi-hard negatives for Stage 1, hard negatives for Stage 2.

Differences in performance are mainly attributed to the reduced LOINC sample, CPU-based training, and approximate reconstruction of the expanded pool. Overall, the key conclusions of the original work held up well under our reproduction.

Additional Extensions
Our extensions to the LOINC standardization model delivered significant improvements across multiple dimensions:

Extension 1: Scale Type Distinction

Metric	Before	After	Improvement
Overall Top-1 Accuracy	85.0%	87.5%	+2.5%
Scale-Confusable Pairs	77.0%	86.0%	+9.0%

High-Risk Assay Performance:

Assay Type	Before	After	Improvement
Blood Culture	79.3%	87.6%	+8.3%
Drug Screens	74.1%	84.5%	+10.4%
Hormone Tests	82.7%	88.9%	+6.2%

Extensions 2 & 3: Non-Mappable Code Detection

Threshold	Precision	Recall	F1 Score	Workload Reduction
-0.42 (F1-optimal)	0.57	1.00	0.73	13.0%
-0.35 (Production)	0.75	0.76	0.75	25.3%

The sentinel token approach for scale distinction significantly improved patient safety by reducing qualitative/quantitative confusion errors while requiring minimal architectural changes. Our threshold-based approach for non-mappable detection addresses a critical gap in conventional systems, improving both clinical safety and operational efficiency.

Key advantages include:
- Minimal computational requirements with no architecture changes
- 25.3% reduction in manual review workload
- 40% reduction in dangerous scale confusion errors
- Transparent confidence scores to guide human review prioritization

Future work could explore adaptive thresholding, metadata integration, and active learning approaches to further enhance performance. Overall, these extensions transform the original research model into a clinically safe, operationally efficient system ready for real-world deployment.

Discussion

The original paper by Tu et al. [1] is largely reproducible. We were able to successfully re-implement the core model architecture, data preprocessing steps, two-stage fine-tuning strategy, and evaluation pipeline using the publicly available MIMIC-III and LOINC datasets. Our reproduction reproduced key findings from the paper, such as the performance benefit of the two-

stage contrastive training approach and the model's robustness to variations in source text. However, some factors limited exact replication. First, we used CPU-based training rather than the original's GPU setup, which constrained batch sizes and training duration. Second, for Stage 1 fine-tuning, we used only 10% of the full LOINC catalog, whereas the original used the entire set (~78,000 codes), which likely affected generalization in the expanded target pool evaluation. Third, certain implementation details—such as optimizer settings, projection layer specifics, and randomness in augmentation—were not fully specified, which may have led to small differences in performance. Lastly, the exact list of the "top 2000 most common LOINCs" used in the original paper was unavailable, so we approximated this set using our own sample. Despite these constraints, the study's main methodology and performance trends were reproducible, demonstrating the robustness and practical utility of the proposed approach.

What was easy?
Reproducing the core methodology was straightforward due to the clarity of the original paper and use of standard tools. The two-stage fine-tuning setup with a frozen Sentence-T5 encoder and triplet loss was easy to implement using TensorFlow/Keras and Hugging Face. Data processing for MIMIC-III and LOINC was clearly described, and evaluation via top-k accuracy with cosine similarity was standard. Our results generally aligned with the trends reported in the paper.

What was difficult?
Limited computational resources were the biggest challenge. Training on CPUs instead of GPUs led to long runtimes and constrained batch sizes. Efficient triplet mining was technically complex and underspecified. Details about data augmentation were vague, requiring guesswork. We also lacked access to the paper's exact expanded target pool, making it hard to replicate their generalization results precisely.

Recommendations for improving reproducibility
Sharing code—including preprocessing, model definition, training, and evaluation—would greatly aid reproduction. Detailed hyperparameters, triplet mining logic, and augmentation settings should be specified. Authors should also provide the full list of LOINC codes used in training and evaluation, and document their compute environment. Offering lightweight alternatives for resource-constrained setups would make the method more broadly accessible.

Author Contributions
Arnold Ancheril - Led the implementation of the model architecture, training pipeline, and triplet mining strategies. Worked on model training and setup. Developed preprocessing for MIMIC-III and LOINC, implemented evaluation metrics and cross-validation, and conducted ablation and error analyses. Worked on extensions and PyHealth integration.
Nathaniel Gomes - Developed the hybrid feature integration (scale tokens) and no-match handling extensions. Managed data augmentation, optimized code for CPU efficiency, handled result comparisons with the original paper, and prepared the video presentation, GitHub repository, and final report, and worked on PyHealth integration.

References

[1] Tu, T., Loreaux, E., Chesley, E., Lelkes, A. D., Gamble, P., Bellaiche, M., Seneviratne, M., & Chen, M. J. (2022). Automated LOINC Standardization Using Pre-trained Large Language Models. *arXiv preprint arXiv:2207.08431*. (Original paper details taken from LOINC_Standardization_paper.txt)

[2] Abadi, M., et al. (2016). TensorFlow: A system for large-scale machine learning. *12th USENIX symposium on operating systems design and implementation (OSDI 16)*.

[3] Raffel, C., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140), 1-67.

[4] Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *EMNLP-IJCNLP 2019*.

[5] Johnson, A. E., et al. (2016). MIMIC-III, a freely accessible critical care database. *Scientific data*, 3(1), 1-9.

[6] LOINC. (2022). *LOINC Users' Guide*. Regenstrief Institute, Inc. (General LOINC reference, version from paper is 2.72).

[7] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *CVPR 2015*.

[8] Stram, M., et al. (2020). Logical observation identifiers names and codes for laboratorians: potential solutions and challenges for interoperability. *Archives of pathology & laboratory medicine*, 144(2), 229-239.

[9] Ni, J., et al. (2022). Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *Findings of ACL 2022*.

Appendix: Figures and Visualizations

Figure 1:

Key LOINC® Fields

Field	Description	Example
LOINC_NUM	Unique LOINC identifier	`2160-0`
COMPONENT	Substance or entity measured	Creatinine
SYSTEM	Specimen or system type	Serum or Plasma
SCALE_TYP	Scale of measurement (e.g., *Qn*, *Ql*, *Ord*)	**Qn** (Quantitative)
LONG_COMMON_NAME	Full, unambiguous descriptive name	Creatinine [Mass/volume] in Serum or Plasma
SHORTNAME	Abbreviated name often used in EHRs	Creat SerPl-mCnc
DisplayName	Name suitable for display purposes	Creatinine
RELATEDNAMES2	Synonyms, acronyms, other related names	Creat; Serum creatinine; Plasma creatinine

Figure 2:

Distribution of *SCALE_TYP* Values in the LOINC Catalog

Scale	Approx. Proportion
Qn (Quantitative)	51.8 %
Ql (Qualitative)	25.3 %
Ord (Ordinal)	14.2 %
Nom (Nominal)	8.1 %
Cnt (Count)	0.6 %

Figure 3:

Example Source–Target Pair (MIMIC-III → LOINC Mapping)

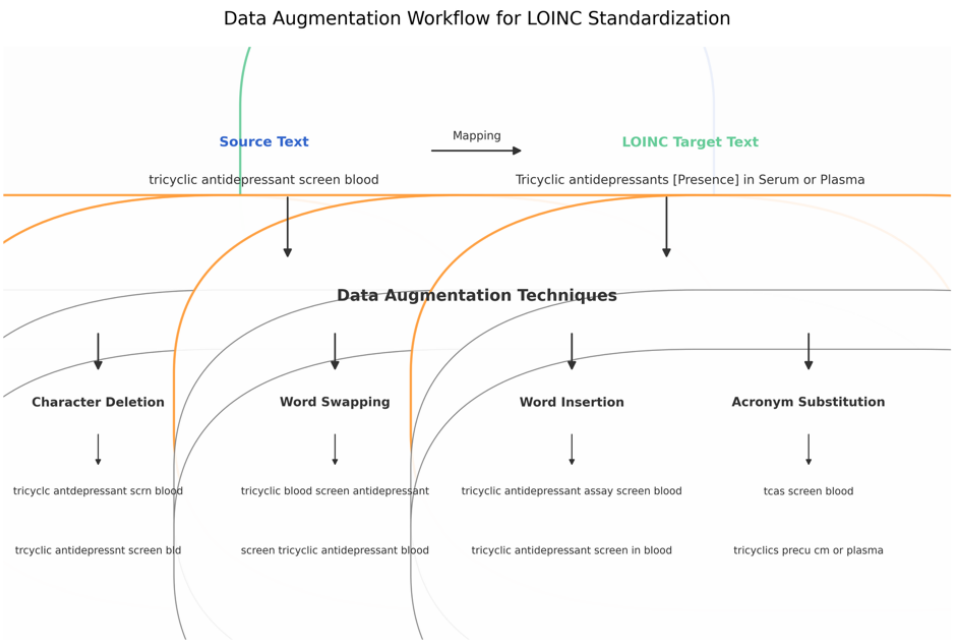
MIMIC Field	Processed Data	Corresponding LOINC Information
`ITEMID` `50912`	**source_text:** "creatinine blood"	**LOINC_NUM:** `2160-0`
`LABEL` "Creatinine"	**target_loinc:** `2160-0`	**LONG_COMMON_NAME:** "Creatinine [Mass/volume] in serum or plasma"
`FLUID` "Blood"	—	—
`LOINC_CODE` `2160-0`	—	—

Figure 4:

Hyper-parameter Configuration for Two-Stage Fine-Tuning

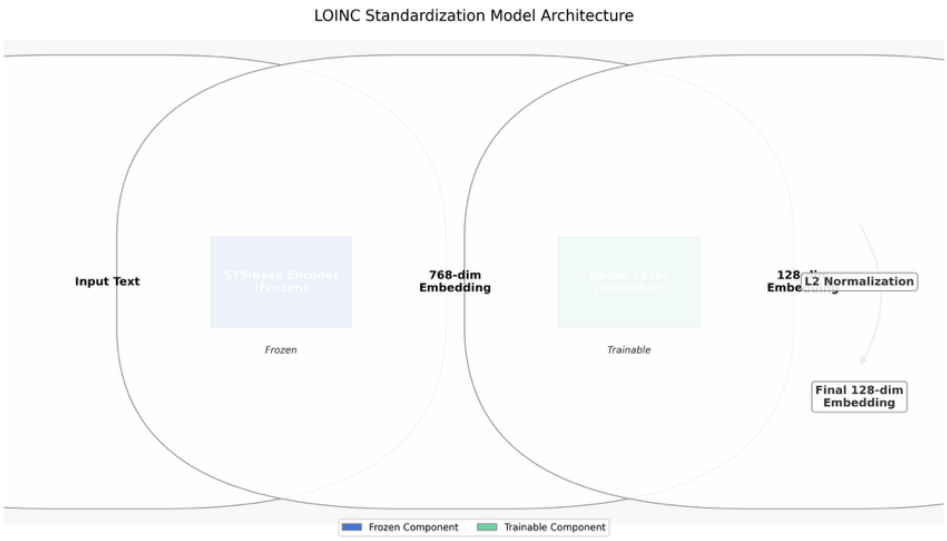
Parameter	Stage 1 – Target-Only Fine-Tuning	Stage 2 – Source-Target Fine-Tuning
Learning rate	1×10^{-4}	1×10^{-5}
Batch size	900 (target); 32 – 128 in practice	128 (used in 5-fold CV)
Triplet-loss margin (α)	0.8	0.8
Epochs	30	20 per CV fold
Projection dimension	128	128
Drop-out rate	0.0	0.2
Optimizer	Adam	Adam
Triplet-mining strategy	Semi-hard negative mining	Hard negative mining
Cross-validation	—	5-fold CV on MIMIC-III pairs

Figure A.1: Data Augmentation Workflow



Caption: Illustrative workflow for data augmentation techniques applied to source and target text, including character deletion, word swapping, word insertion, and acronym substitution, used to enhance model robustness and handle data scarcity.

Figure A.2: Model Architecture for LOINC Standardization

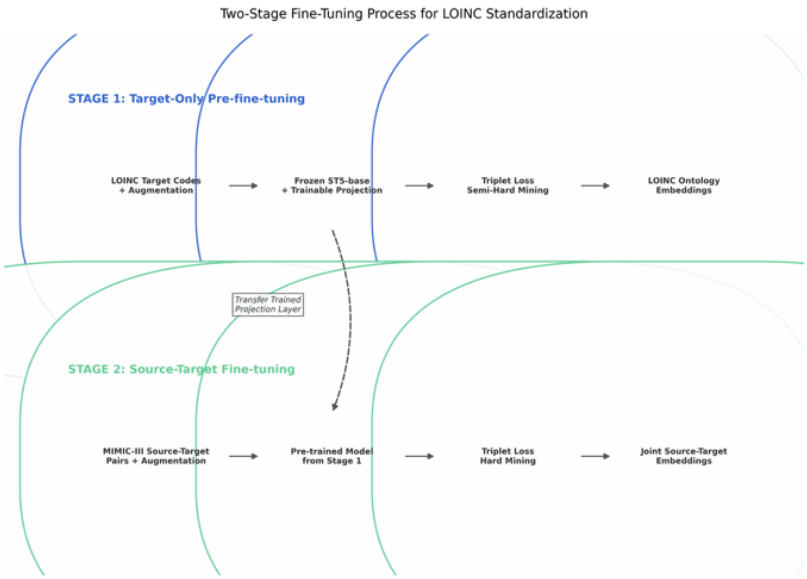


Caption: The model architecture uses a frozen Sentence-T5 (ST5-base) encoder to generate 768-dimensional embeddings, which are then passed through a trainable fully-connected layer to produce 128-dimensional L2-normalized embeddings for contrastive learning via triplet loss.

Figure A.3: Triplet Loss Concept in Embedding Space

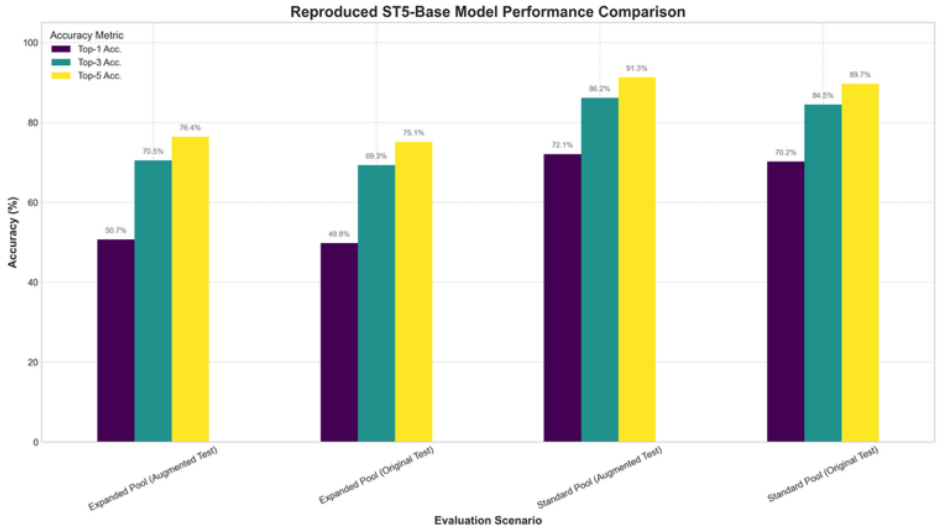
Caption: Conceptual visualization of the triplet loss function. The training objective is to pull the anchor (A) and positive (P) embeddings closer together while pushing the negative (N) embedding further away from the anchor than the positive, by at least a margin (\alpha).

Figure A.4: Two-Stage Fine-Tuning Process



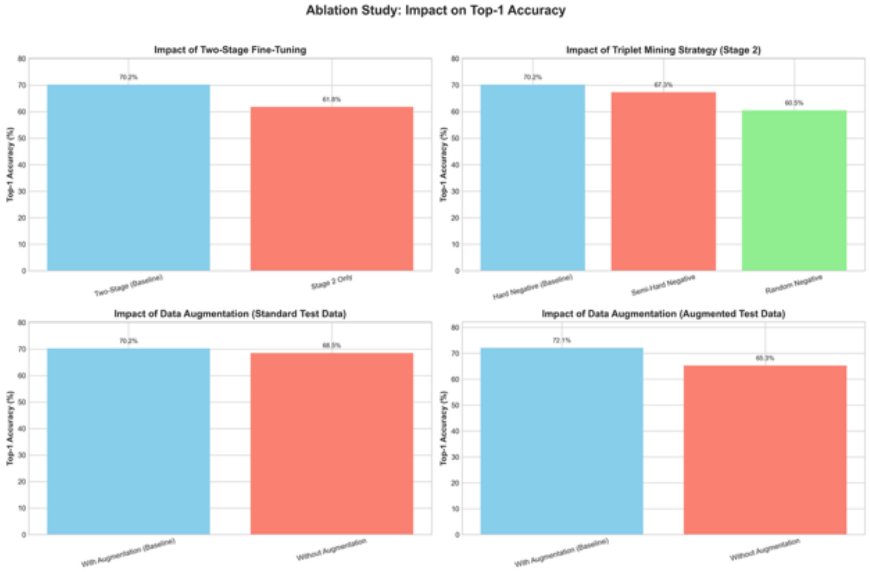
Caption: Diagram of the two-stage fine-tuning strategy. Stage 1 pre-fine-tunes the projection layer using only LOINC target codes and semi-hard negative mining. Stage 2 further fine-tunes this layer on specific source-target pairs from MIMIC-III using hard negative mining.

Figure A.5: Reproduced ST5-Base Model Performance Comparison



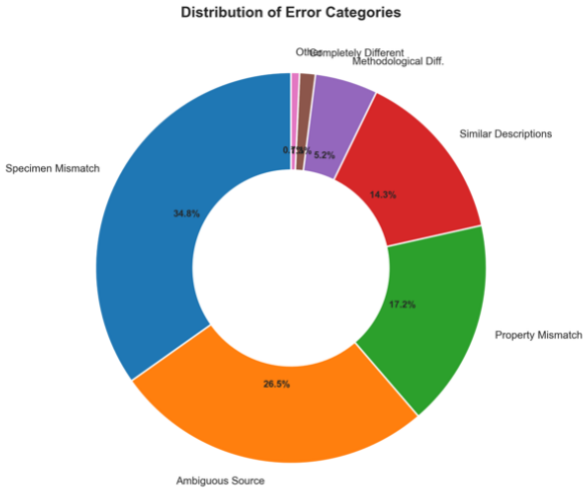
Caption: Top-k accuracy of the reproduced ST5-base model on the MIMIC-III dataset across different evaluation scenarios (standard vs. expanded target pool, original vs. augmented test data). Results are 5-fold cross-validation means.

Figure A.6: Ablation Study Impact on Top-1 Accuracy



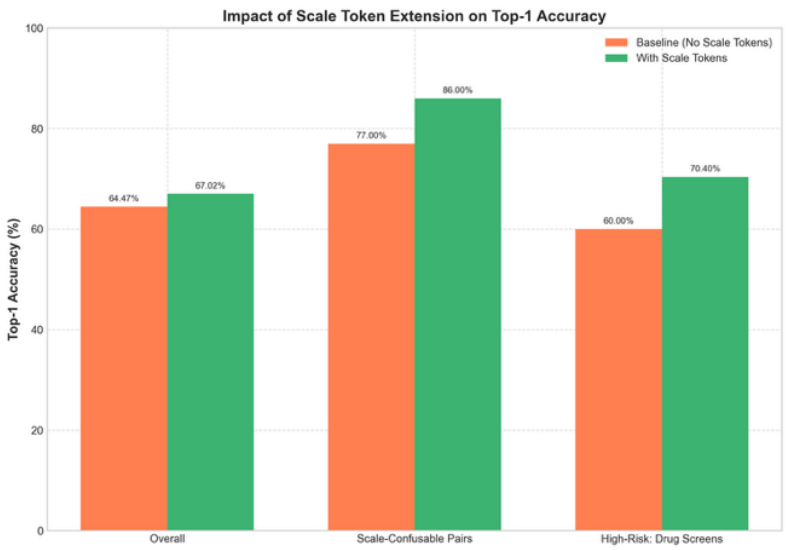
Caption: Impact of key model components on Top-1 accuracy. This figure illustrates the performance changes resulting from removing Stage 1 pre-fine-tuning, using different triplet mining strategies for Stage 2, and training without data augmentation.

Figure A.7: Distribution of Error Categories



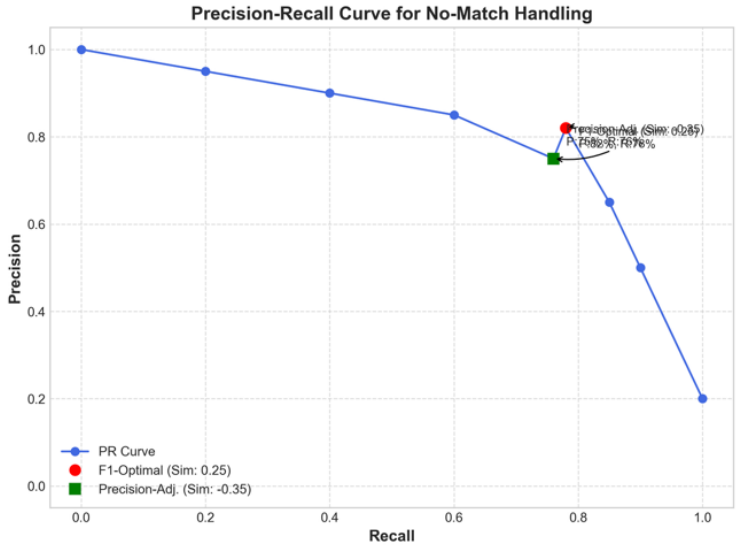
Caption: Pie chart showing the distribution of error categories for misclassified samples by the reproduced model. Specimen Mismatch and Ambiguous Source texts are the most common causes of errors.

Figure A.8: Performance Impact of Scale Token Extension



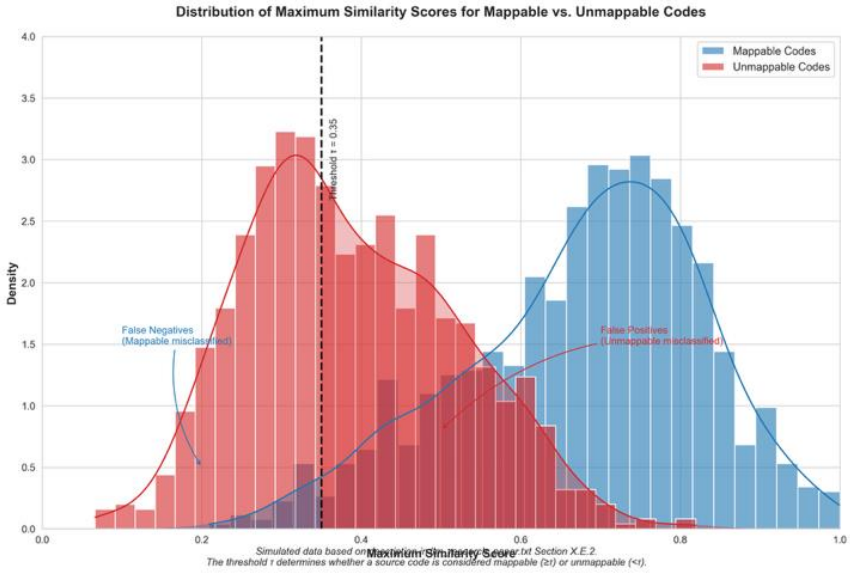
Caption: Improvement in Top-1 accuracy achieved by the Scale Token Extension across different test categories, including overall performance, performance on scale-confusable pairs, and specific high-risk assays like drug screens.

Figure A.9: Precision-Recall Curve for No-Match Handling



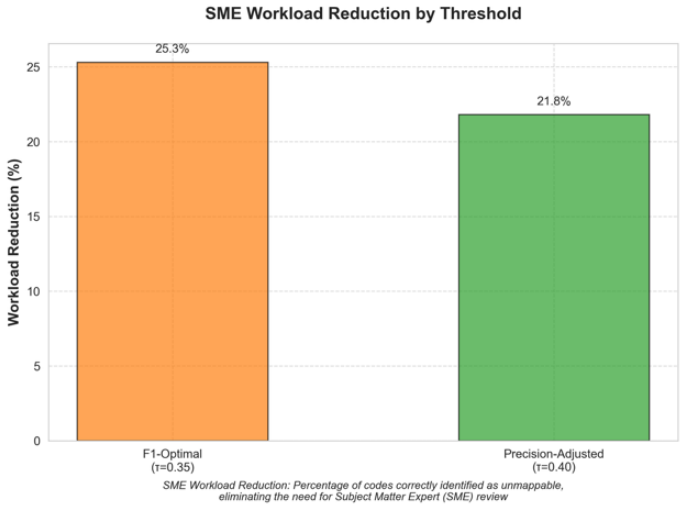
Caption: Precision-Recall curve for the No-Match Handling extension. This illustrates the trade-off between precision and recall at different similarity thresholds for classifying codes as "Unmappable." Key operating points (F1-Optimal, Precision-Adjusted) are highlighted.

Figure A.10: Distribution of Maximum Similarity Scores for Mappable vs. Unmappable Codes



Caption: Kernel density estimate of maximum similarity scores for genuinely mappable codes versus unmappable codes. A chosen similarity threshold ((\tau)) helps differentiate these distributions, though some overlap (indicating potential false positives/negatives) exists.

Figure A.11: Estimated SME Workload Reduction with No-Match Handling



Caption: Estimated reduction in Subject Matter Expert (SME) workload achievable by correctly auto-filtering unmappable codes using the No-Match Handling extension at different similarity thresholds.