LLM-STARS: LLM-Enhanced Standardization of Time-series Analysis and Relationships in Subledgers

Wei Tang, Daksha Yadav, Xiaoli Zhang, Boyang Tom Jin
Amazon
twtang97@gmail.com, {dakyadav, zhasabri, boyanjin}@amazon.com

ABSTRACT

Financial accounting systems rely heavily on subledgers to track detailed transaction records. However, modern systems often evolve into complex architectures where different components use inconsistent labeling conventions, making it difficult to understand and utilize important relationships within subledger data. This paper presents a novel framework LLM-STARS (LLM-Enhanced Standardization of Time-series Analysis and Relationships in Subledgers) that leverages Large Language Models to enhance time series analysis of subledger data through relationship modeling. LLM-STARS represents subledger data as a graph where financial events connect accounting segments, while firstly using LLMs to generate standardized interpretations of these events based on both their attributes and their role in moving money through the accounting system and then explicitly modeling relationships among subledger time series. The framework effectively identifies two types of relationships between subledger activities: reconciliation relationships that capture clearing/settlement patterns, and similar pattern relationships that reflect shared business drivers. We demonstrate through extensive experiments on enterprise testing data representative of real-world usage patterns that incorporating these relationships significantly improves the subledger data analysis performance as compared to traditional univariate approaches. For example, LLM-STARS improves anomaly detection F1 score from 0.516 to 0.621 (by 20.3%) for collective seasonal outliers and decreases symmetric mean absolute percentage error from 53.82 to 27.83 (by 48.3%). Moreover, LLM-STARS provides interpretable results through with language descriptions while maintaining the technical rigor required for financial applications.

KEYWORDS

Large Language Models, Representation Learning, Data Mining, time series Forecasting

ACM Reference Format:
Wei Tang, Daksha Yadav, Xiaoli Zhang, Boyang Tom Jin . 2025. LLM-STARS: LLM-Enhanced Standardization of Time-series Analysis and Relationships in Subledgers. In *KDD 2025 Workshop on Machine Learning in Finance, August 03–07, 2025, Toronto, CA.* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

KDD Workshop on MLF, August 03–07, 2025, Toronto, CA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

General ledger (GL) balances represent aggregated financial positions for various accounting elements (e.g., assets, liabilities, etc.). The inherent aggregation within the GL necessitates supplementary data structures to provide the requisite level of transactional specificity for operational and analytical purposes. Subledgers address this requirement by providing a detailed breakdown of individual transactions that collectively comprise the summary balances reported in corresponding GL control accounts. The architectural relationship can be conceptualized as a one-to-many mapping, where a single GL control account (e.g., Accounts Receivable) is supported by numerous individual records within its associated subledger. As illustrated in Figure 1, each subledger entry contains three distinct categories of attributes:

(1) **Source Attributes (or Financial Event Attributes):** These capture the origin and nature of transactions, including business activities (such as sales or payments) from business events and system-generated events (such as periodic revenue recognition or accruals) from accounting regulations.
(2) **Accounting Segment Attributes:** These define the structural organization within the ledger system, including account numbers and classifications that distinguish between different types of subledgers (e.g. Receivables, Inventory).
(3) **Common Informative Attributes:** These provide essential transaction details such as dates, amounts, and currency codes.

Subledger data contains crucial relationships that reflect the flow of financial activities. These relationships manifest in two key ways: First, as direct reconciliation, where one type of transaction in the subledger clears or nets out another entirely or partially. For example, as illustrated in Figure 1, a sale event creates an entry in the Customer Advance segment, which is later cleared by a shipment event. This shipment event simultaneously creates entries in the Revenue and Tax segments, representing the recognition of income and associated taxes when the goods are delivered. Second, as similar patterns, where different subledger use cases share similar temporal behaviors due to similar underlying business natures. For instance, Figure 2 shows how sale events in Customer Advance accounts across different markets or product lines often exhibit similar trend and seasonal patterns, reflecting shared economic drivers or consumer behaviors. Understanding and using these relationships is essential for effective subledger data analysis. Figure 2 illustrates how these relationships can improve time series analysis: the related time series share similar temporal patterns and could help improve time series prediction and anomaly detection.

However, modern financial accounting systems often evolve into complex, decentralized architectures where different components are managed by separate business teams using inconsistent labeling

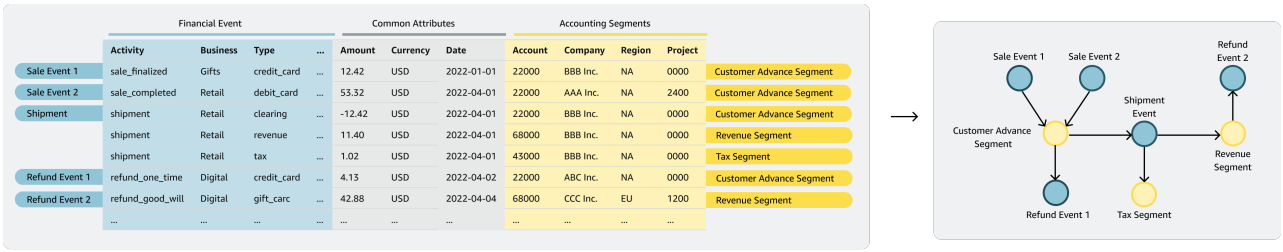

Figure 1: Illustration of subledger activity representation: (a) A sample table showing raw subledger transaction data with non-standardized financial event attributes (in blue columns) and accounting segment attributes (in yellow columns). b) The same data represented as a graph where financial events (blue circles) connect to accounting segments (yellow circles). The graph structure reveals how money flows through the accounting system - for example, sale events connect customer advance segments to revenue segments, while refund events flow in the opposite direction. Note that the attribute names, values, and labels are simplified for illustration purposes and do not reflect any real data.

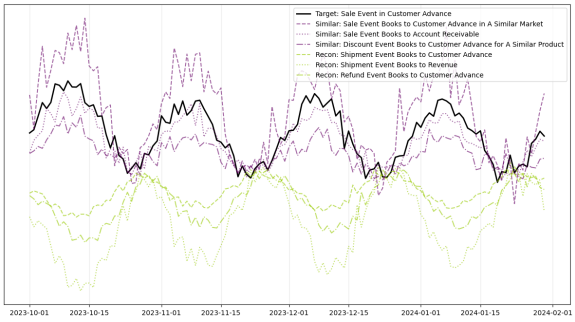

Figure 2: Illustration of Subledger Use Case Relationships: Similar and Reconciling relationships can improve subledger time series analysis.

and booking conventions.[1] While accounting segment attributes adhere to a strictly defined Chart of Accounts (COA)[2], financial event attributes often lack consistency due to the complex nature of the system. In large organizations, this complication can result in a vast quantity (millions in some cases) of configuration rules managed by hundreds of different teams, generating more than hundreds of thousands distinct subledger use cases. This extensive scale of non-standardization poses significant challenges for understanding and utilizing the important relationships within subledger data. Additionally, although accounting experts can help label and document those relationships for some example case, it is almost impossible to scale. An automated approach to understand subledger data and mining relationship is needed.

As illustrated in Figure 1, the non-standardization of financial events manifests in different ways. Take one example: similar activities may be recorded with inconsistent attribute combinations. For instance, the same sale completion events could be recorded with different styles: one business subsidiary might record a sale event as "sale_finalized" in the Financial Activity field, while another subsidiary might use "sale_completed". Other non-standardization includes attribute names may be overloaded and used to record different information; or repeative attributes to record the same information.

While financial events may be labeled differently across accounting systems, they share a fundamental commonality: they all represent the movement of money between accounting segments. Through double-entry accounting principles, these monetary movements carry specific and consistent meanings that help explain both individual financial events and their relationships to one another.

Large Language Models (LLMs) such as Claude[3] have demonstrated remarkable capabilities in understanding domain-specific concepts and synthesizing information from various sources[17]. Those capabilities can help standardize the financial events as well as uncover common relationship among them.

Leveraging these insights, we propose a unified framework: LLM-Enhanced Standardization of Time-series Analysis and Relationships in Subledgers (LLM-STARS) to first understand subledger data, mining relationships among subleger use cases and then achieve better time series analysis results. Our work makes the following key contributions:

(1) We propose a novel graph representation of subledger data and develop an LLM-based framework that leverages this structure to generate standardized text representations for subledger use cases.
(2) We introduce a systematic approach to identify and utilize two types of relationships in subledger data: reconciliation relationships that capture clearing/settlement patterns, and similar pattern relationships that reflect shared business drivers.
(3) We conduct extensive experiments on real-world subledger data from a large company's testing environment, demonstrating that our relationship-aware approach significantly improves subledger activity forecasting and anomaly detection performance.

[1]Possible Driven Factors: 1) Multiple ERP and Sub-ledger Systems working together due to merger and acquisition 2) Heterogeneous business models and revenue streams require different booking logic, accrual rules. 3) Different tax rules and accounting regulations in different countries 4) Evolving accounting architectures to accommodate those dynamic complexities.
[2]https://en.wikipedia.org/wiki/Chart_of_accounts

[3]https://www.anthropic.com/claude

2 RELATED WORKS

Modeling Relationships in Time Series. Time series analysis incorporating relational information has been an active research area. Recent approaches can be broadly categorized into those leveraging deep learning to implicitly model relationships and those explicitly modeling relationships using graphs.

Within deep learning, encoder-decoder architectures with specialized neural network modules are common. For instance, TimeXer [16] employs carefully designed attention mechanisms to learn inter-series relationships from exogenous time series, while CrossFormer [22] utilizes a two-stage attention layer to capture inter-series relationships alongside customized encoder and decoder components. Although modeling relationships in a hidden embedding space has proven effective for downstream tasks, these methods often suffer from limited explainability.

Another significant research direction directly models relationships using graphs. Connections in these graphs can be constructed heuristically or learned end-to-end with downstream tasks, as detailed in the survey by [5]. Causal graphs, a specific type of graph also known as Bayesian networks, represent conditional distributions among variables. Examples include Dynotears[14], which learns causal dependencies among different time series and time lags through score-based optimization, and the approach in [20], which transforms anomaly detection into a causal graph learning problem to identify anomalies based on disrupted causal relationships. Beyond learned relationships, domain knowledge can also drive effective models. In cases where relationships are well understood, directly incorporating this knowledge can lead to strong performance. MechBayes[3]'s success in the US COVID-19 Forecast Hub[1] competition, where a Bayesian extension of the classic epidemiological SEIR model outperformed complex deep learning methods by leveraging known disease transmission patterns, exemplifies this.

Finally, in the domain of accounting ledger analysis, researchers have also explored relational modeling. [19] utilizes a Transformer-based architecture for anomaly detection with explicit relationship modeling, and [4] learns general ledger representations using GNNs with graphs defined by double-booking entries. However, these approaches do not fully address the challenges posed by non-standardized financial event attributes.

LLM and Domain Knowledge Extraction. Large Language Models (LLMs) demonstrate emergent abilities [17] to perform various tasks beyond their initial training objectives, including domain-specific applications. This capability creates opportunities to leverage domain knowledge from general-purpose LLMs for specialized tasks. One line of research focuses on utilizing LLMs to construct knowledge graphs for domain-specific information, which can then be applied to downstream tasks [2, 9]. A comprehensive overview of LLM applications in various information extraction tasks is presented in [18]. Another research direction explores fine-tuning or knowledge distillation techniques to enhance domain-specific capabilities in open-source or smaller LLMs. For instance, [21] demonstrates fine-tuning an LLM for legal applications, while [12] develops a specialized LLM for time series anomaly detection through knowledge distillation from a general-purpose model.

LLM and Time Series. Recent progress in LLMs has demonstrated their strong capabilities in Natural Language Processing and Computer Vision, sparking interest in their potential for time series analysis. However, a key challenge lies in the fact that current LLMs are not trained on extensive time series datasets. Consequently, their direct application to time series tasks often under-performs compared to specialized, state-of-the-art models designed for this domain.

To address this limitation, several approaches have emerged. One strategy involves adapting LLM architectures to enable direct reasoning over time series data. For example, TimeLLM [6] introduces input reprogramming, effectively mapping time series data into the textual embedding space without altering the core LLM architecture. This method achieves competitive performance with minimal parameter adjustments. Nevertheless, these modified architectures can still demand considerable computational resources, especially when compared to utilizing readily available, pretrained LLMs. An alternative line of research explores leveraging the inherent knowledge extraction abilities of LLMs to enhance downstream time series tasks. RealTCD [11] exemplifies this by employing LLMs to initialize causal relationship matrices for temporal causal discovery in IT operations data. For a comprehensive survey of these diverse strategies aimed at bridging the gap between time series analysis and LLMs, readers are referred to [7, 8].

3 PRELIMINARIES

3.1 Dataset Description

In this paper, we analyze daily subledger activity summaries derived from enterprise testing data reflecting real-world usage. While the monetary amounts differ from production data, this dataset preserves key characteristics of the actual system: the data volume, the complexity of financial attributes, and the intrinsic relationships between financial events and accounting segments.

Each record in the activity summary dataset represents a **subledger use case** - a unique combination of a financial event and an accounting segment, along with its daily aggregated monetary value. As visualized in Figure 1, these use cases can be represented as edges in a graph, where each edge captures the flow of money between corresponding financial event and accounting segment. The activity summary table tracks these monetary flows on a daily basis, creating a time series for each use case.

In this paper, we focus on the subledger dataset spanning 13 months related to a specific accounting area. The dataset comprises:

- Approximately 20,000 subledger use cases
- Around 1,000 financial events
- Approximately 3,000 accounting segments

We refer to this dataset as "SubledgerVirtualEnv" in the rest of the paper.

3.2 Problem Formulation

Definition 3.1 (Subledger Use Case). A subledger use case S is defined by a unique combination of financial event F_s and accounting segment A_s. Formally, $S = (F_s, A_s)$ where:

- F_s represents the financial event, categorized by a set of related attributes (e.g., activity, business unit, component)

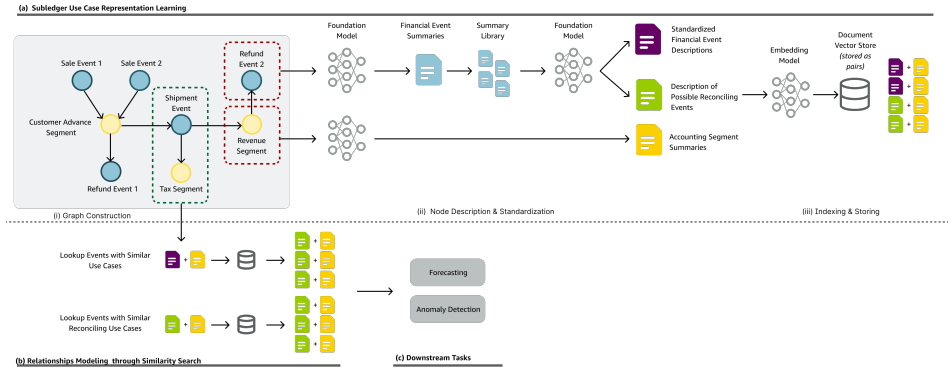

Figure 3: Overview of LLM-STARS framework. (a) First, represent subledger activity summary as a money flow graph and generate standardized text representations for financial events and accounting segments. (b) Next, we model two types of relationships between subledger use cases: similar pattern relationships that capture shared business drivers, and reconciliation relationships that capture clearing/settlement patterns. (c) Finally, we utilize these identified relationships to enhance downstream tasks including time series forecasting and anomaly detection.

- A_S represents the accounting segment, categorized by a set of COA attributes (e.g., account code, company)

Definition 3.2 (Subledger Use Case Activity Time Series). For each subledger use case s, its time series $X_s = \{x_{s,t}\}_{t=1}^{T}$ represents the daily aggregated transaction amounts, where $x_{s,t}$ is the total amount on day t.

Given a set of subledger use cases $S = \{s_1, ..., s_n\}$ and their corresponding time series $X = \{X_{s_1}, ..., X_{s_n}\}$. Our goal is to achieve better time series prediction and anomaly detection through mining the relationships within S.

4 PROPOSED LLM-STARS METHOD

In this section, we describe the proposed LLM-STARS (LLM-Enhanced Standardization of Time-series Analysis and Relationships in Subledgers) framework. The framework consists of three steps: 1) Generate accounting domain knowledge-rich text representations for subledger use cases by modeling subledger activity summary as a graph and utilizing LLMs to generate standardized representations from the graph structure; 2) Use these representations to explicitly model two types of relationships among subledger use cases - similar pattern relationships that capture shared business drivers, and reconciliation relationships; 3) Leverage these relationships to perform enhanced subledger time series analysis including forecasting and anomaly detection. A visualization of LLM-STARS framework can be seen in Figure 3.

4.1 LLM Based Subledger Use Case Representation Learning

4.1.1 Graph Construction. As illustrated by Figure 1, we represent subledger use cases a graph $G = (V_F \bigcup V_A, E, W)$ where:

(1) V_F represents financial event nodes.
(2) V_A represents accounting segment nodes.
(3) $E \subseteq V_F \times V_A$ represents the set of edges. In this definition, each edge naturally corresponding to one subledger use case $e = s = (f_s, a_s)$.
(4) $W : E \rightarrow \mathbb{R}$ assigns weights to edges, where for each subledger use case s: $W(s) = \sum_{t=t_0}^{t_0+N} x_{s,t}$, where $x_{s,t}$ is the transaction amount in subledger use case s on day t. The sign of $W(s)$ indicates the direction of money movement: positive values represent money flowing from financial events to accounting segments, while negative values indicate the reverse flow.

4.1.2 Context Collection. For each node $v \in V_F \cup V_A$, we collect context $C(v)$ from its local neighborhood $N(v)$:

$$C(v) = attr(v) \cup \bigcup_{u \in N(v)} \{attr(u), edge(v, u)\} \tag{1}$$

where $attr(v)$ represents the node's attributes and partial internal description, and $edge(v, u)$ contains the edge information (amount and direction) between nodes v and u.

For computational efficiency, when $|N(v)| > k$ (where k is a threshold), we sample k neighbors with probability proportional to $\log(|W(e)| + 1)$, where $W(e)$ is the edge weight and the offset of 1 ensures positive sampling weights.

4.1.3 Node Description and Label Generation. We employ a three-stage process to generate standardized descriptions and labels:

(1) **Node Description Generation:** Generate comprehensive descriptions by aggregating neighborhood information:

$$S(v) = LLM(C(v), \{C(u) : u \in N(v)\}) \tag{2}$$

where $S(v)$ is a natural language description incorporating both the node's context $C(v)$ and its neighbors' contexts $\{C(u)\}$. This semantic aggregation is analogous to neighborhood aggregation in GCNs, but produces textual descriptions rather than numerical embeddings.

(2) **Label Space Generation:** Create a standardized set of labels with descriptions:

$$L = \{(l_i, D(l_i))\}_{i=1}^{k} = LLM(\{S(v) : v \in V_F \cup V_A\}) \tag{3}$$

where L represents the unified label space, with each label l_i paired with its standard description $D(l_i)$. This step identifies common patterns across node descriptions, similar to clustering centers or topic themes. For computational efficiency with large numbers of financial event nodes, we employ a divide-and-conquer approach to merge similar descriptions and generate the label space. Details can be see in the Appendix A.1.

(3) **Label Assignment:** Map nodes to appropriate labels:

$$label(v) = LLM(S(v), \{(l_i, D(l_i))\}_{i=1}^{k}) \in L \tag{4}$$

where each node is assigned a label from the unified space based on semantic similarity between its description $S(v)$ and the standard label descriptions $D(l_i)$.

Detailed prompt examples can be seen in Appendix A.2. Appendix A.3 shows an example of the final standardized text representation of a financial event.

4.2 Explicit Subledger Use Case Relationship Modeling

With standardized representations of financial events and accounting segments, we can explicitly model relationships among subledger use cases to enhance time series analysis. For each use case $s = (f_s, a_s)$, we combine the semantic descriptions of its financial event node and accounting segment node to identify meaningful relationships with other use cases.

4.2.1 Related Time Series Identification. We identify two types of relationships between subledger use cases:

(1) **Reconciliation Relationships:** These capture pairs of financial events that typically clear or settle each other.
(2) **Similar Pattern Relationships:** These identify events that exhibit similar behavioral patterns due to shared business drivers.

For similar pattern relationships, we search semantically similar use cases using embedding-based search:

$$R_{similar}(s) = \text{TopK}(\{s_i | sim(s, s_i), s_i \in S_{all} \setminus \{s\}, k) \tag{5}$$

Reconciliation relationships cannot be directly searched using the original event representation. We have to generate the representations of the potential reconciliation events to search them. In other words, we need to get a text description for possible reconciliation events. This is done by a LLM call. Prompt example can be seen in Appendix A.2. Therefore, we apply a two-step approach:

$$R_{recon}(s) = \text{TopK}(\{s_i | sim(s_i, D_{recon}(s)), s_i \in S_{all} \setminus \{s\}\}, k) \tag{6}$$

where $D_{recon}(s)$ is an LLM-generated description of events that could potentially reconcile with s (Appendix A.4 shows one example). Topk similar search is achieved by embedding search of the text representations of the query and candidate. Encoder can be any encoder that support long text embeddings.

4.3 Relationship Enhanced Subledger Time Series Analysis

4.3.1 Time Series Forecasting. We leverage these identified relationships to enhance time series forecasting:

$$\hat{y}_s(t) = f(R_{recon}(s), R_{similar}(s), t) \tag{7}$$

where $\hat{y}_s(t)$ is the forecasted activity for use case s at time t, and f is a forecasting function that incorporates information from related time series.

In our implementation, we utilize Facebook Prophet [15] as our base forecasting model, which decomposes time series into trend, seasonality, and holiday components. Prophet is renowned for its interpretability and scalability, making it an ideal foundation model for evaluating the subledger relationships identified in the previous section. We incorporate the reconciliation and similarity relationships as exogenous variables in the Prophet model, allowing it to capture both accounting-specific patterns and broader behavioral similarities.

4.3.2 Anomaly Detection. We detect anomalies by comparing prediction errors normalized by the width of the prediction interval:

$$A(s,t) = \frac{|y_s(t) - \hat{y}_s(t)|}{\hat{u}_s(t) - \hat{l}_s(t)} \tag{8}$$

where $y_s(t)$ is the actual value at time t; $\hat{y}_s(t)$ is the point prediction; $[\hat{l}_s(t), \hat{u}_s(t)]$ is the prediction interval.

An anomaly is flagged when $A(s,t) > \tau$. This approach normalizes the prediction error by the model's uncertainty (represented by the prediction interval width), meaning that a given prediction error is considered more significant when the model is more certain (narrow prediction interval) than when it is less certain (wide prediction interval).

5 EXPERIMENTAL ANALYSIS

Due to the lack of labeled data, we cannot directly evaluate the quality of LLM-generated text representations and identified relationships between subledgers. Instead, we perform an indirect evaluation by measuring the effectiveness of these representations and relationships when applied to two downstream tasks mentioned in section 4.3. This section presents our experimental methodology, evaluation metrics, and empirical results.

5.1 Implementation Details

5.1.1 Backbone Models. LLM-STARS is designed to be compatible with various large language models. In this implementation, we utilize Anthropic's Claude 3.7 Sonnet model[4] for LLM related operations. For encoding node descriptions and potential reconciliation event descriptions, we employ the titan-embed-text-v2[5]. However, it's worth noting that our framework allows for the use of alternative encoders as needed.

5.1.2 Time series Forecasting. We evaluate forecasting performance of LLM-STARS using 13 months' data from the "SubledgerVirtualEnv" dataset described in section 3.1. Each experiment uses the first 12 months as the training period and the last month as the evaluation period. The forecasting problem is framed as a multi-horizon task, where the horizon length equals the number of days in the evaluation month. After filtering out the use cases that does not

[4]https://www.anthropic.com/news/claude-3-7-sonnet
[5]https://huggingface.co/amazon/Titan-text-embeddings-v2

have enough historical data for analysis, our experiments includes approximately 4,000 use cases.

We utilize Prophet python package[6] to implement the forecasting and anomaly detection. We configure all models with both weekly and monthly seasonality components enabled while maintaining default values for all other parameters. We model relationships between subledger use cases by incorporating related time series as additional regressors in the Prophet model. During forecasting, the future values of these related time series are made available to the model, allowing it to generate predictions conditioned on the expected behavior of related activities.

5.1.3 Anomaly Detection. Since we employ prediction-based anomaly detection, it is a natural extension of the time series forecasting setup. However, due to the inherent scarcity of labeled anomaly data, we employed a data synthesis approach to generate anomalous data points for our experiments. Following [10], we created synthesized point-wise and pattern-wise anomalies. Point-wise anomalies consist of single outliers with respect to their local and global time series data, while pattern-wise anomalies include outliers that exhibit unexpected changes in seasonality, shape, and trends. We generate anomaly data for all five scenarios, perform anomaly detection, and report metrics for each scenario separately. We calculate the anomaly score as described in Section 4.3.2 and report anomalies at the point level with thresholds ranging from 0.001 to 50.

5.2 Baselines And Ablations

To evaluate the effectiveness of LLM-STARS framework and analyze the contribution of each component, we compare against the following baseline methods and conduct ablation studies:

5.2.1 Baseline Methods:

(1) **Univariate Forecasting:** A traditional approach that treats each subledger use case independently, using only its own historical data for forecasting. This serves as our primary baseline to demonstrate the value of incorporating relationship information.
(2) **Time Series Similarity-Based:** This method identifies related time series through direct similarity computation (cosine similarity) on historical time series, without utilizing any semantic information. This baseline helps evaluate the benefit of our LLM-based relationship identification approach compared to purely statistical methods.

5.2.2 Ablation Studies:

(1) **No Neighborhood Information:** Removes the neighborhood context during node description generation, using only the node's own attributes. This ablation tests the importance of incorporating local graph structure in generating comprehensive node descriptions.
(2) **No Standardization:** Skips the financial event standardization step, using raw descriptions without merging similar events. This helps quantify the impact of our event standardization process on downstream tasks.
(3) **No Reconciliation Relationships:** Only considers similar pattern relationships while excluding reconciliation relationships. This ablation evaluates the specific contribution of accounting-domain-specific reconciliation patterns to the model's performance.

[6] https://facebook.github.io/prophet/

5.3 Evaluation Metrics

5.3.1 Time Series Forecasting. For subledger activity forecasting, we utilize the following commonly used metrics: symmetric mean absolute percentage error (sMAPE), mean absolute scaled error (MASE), and scaled interval score (SIS), which are also employed in popular time series forecasting evaluations such as the M4 competition [13]. Detailed metric definitions are provided in Appendix B. sMAPE and MASE measure point-wise forecasting accuracy, while SIS assesses interval forecasting accuracy. SIS and MASE are scaled by the prediction error of a naive model that assumes the time series exhibits stable seasonality, using the data point from the latest seasonal period as the prediction. Consequently, these metrics are controlled by a parameter S, representing the seasonality lag of the naive model. Given that we lack precise information regarding the seasonality of our data, we choose to evaluate our models using multiple values for S: 1 (no seasonality, using the last seen data point as the naive prediction), 7 (weekly seasonality), and 30 (monthly seasonality). To mitigate the impact of extremely good or bad outliers, we aggregate the metrics using the geometric mean.

5.3.2 Anomaly Detection. For evaluating anomaly detection performance, we employ two complementary metrics: F1 score and Area Under the Curve (AUC) score. To calculate the F1 score, we first determine the optimal threshold on a validation dataset. The AUC score, which is threshold-independent, measures the model's ability to distinguish between normal and anomalous points across all possible threshold values. The combination of both F1 and AUC scores provides a comprehensive evaluation of our anomaly detection system's performance.

5.4 Results

This section presents the empirical evaluation of the proposed LLM-STARS framework on the SubledgerVirtualEnv dataset. We assess its performance on two key downstream tasks: time series forecasting and anomaly detection. The quantitative results for these tasks are summarized in Tables 1 and 2, respectively.

5.4.1 Time Series Forecasting. Table 2 presents the time series forecasting performance of LLM-STARS and the comparative methods on the SubledgerVirtualEnv dataset. We report the symmetric Mean Absolute Percentage Error (sMAPE), Mean Absolute Scaled Error (MASE) with seasonality lags of 1, 7, and 30, and the Multiscale Interval Score (MSIS) with the same seasonality lags. These metrics provide insights into both point forecast accuracy and prediction interval quality.

The results demonstrate that LLM-STARS consistently outperforms all baseline and ablation methods across all forecasting metrics. It achieves the lowest sMAPE (27.8275) and MASE values for all seasonality lags (0.2584, 0.2769, and 0.1849), indicating more accurate point forecasts. Furthermore, LLM-STARS also yields the lowest MSIS values (2.3841, 2.5544, and 1.7057), suggesting better

Table 1: Anomaly Detection Accuracy on SubledgerVirtualEnv Dataset with Synthetic Anomalies. Experiments are done in five anomaly situations. AUC and F1 score reported in each anomaly situation.

anomaly_type	collective_global_outliers		collective_seasonal_outliers		collective_trend_outliers		point_contextual_outliers		point_global_outliers	
metric	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
method										
ablation_no_neighbor	0.721	0.534	0.786	0.617	0.610	**0.616**	0.723	0.503	0.852	0.646
ablation_no_recon	0.721	0.530	0.786	0.615	0.610	**0.616**	0.722	0.502	0.852	0.646
ablation_no_standardization	0.706	0.522	0.775	0.612	0.606	0.615	0.716	0.495	0.846	0.634
baseline_univariate	0.510	0.452	0.628	0.516	0.556	0.610	0.586	0.410	0.786	0.542
baseline_ts_similarity	0.706	0.506	0.773	0.579	0.603	0.612	0.698	0.471	0.847	0.637
Proposed LLM-STARS	**0.723**	**0.535**	**0.789**	**0.621**	**0.612**	0.615	**0.728**	**0.508**	**0.855**	**0.649**

calibrated and more informative prediction intervals. These results indicate that directly leveraging LLM-enhanced representations and relationships provides a more effective foundation for accurate time series forecasting in the subledger context.

5.4.2 Anomaly Detection. Table 1 details the anomaly detection accuracy of LLM-STARS and several baseline and ablation methods across five distinct synthetic anomaly scenarios: collective global outliers, collective seasonal outliers, collective trend outliers, point contextual outliers, and point global outliers. For each scenario, we report the Area Under the Curve (AUC) and F1 score, providing a comprehensive view of the detection performance.

The results clearly indicate that LLM-STARS achieves the highest performance across the majority of anomaly types. Specifically, LLM-STARS yields the best AUC scores in all five anomaly scenarios (0.723, 0.789, 0.612, 0.728, and 0.855) and the highest F1 scores in four out of the five scenarios (0.535, 0.621, 0.508, and 0.649). This consistent superiority highlights the effectiveness of the LLM-enhanced representations and identified subledger relationships in accurately distinguishing anomalous patterns from normal behavior.

In contrast, the baseline methods, including the univariate baseline and the time series similarity-based baseline, generally exhibit lower performance. For instance, the univariate baseline struggles significantly across all anomaly types, demonstrating the importance of considering multivariate relationships captured by LLM-STARS.

The consistent superior performance of LLM-STARS across both anomaly detection and time series forecasting tasks underscores the effectiveness of leveraging LLMs to generate meaningful representations of subledger activity and to model the intricate relationships between them. The ablation studies further validate the contribution of each component within the proposed framework. These empirical results strongly support the potential of LLM-STARS as a robust and versatile framework for analyzing and understanding complex subledger data.

The only exception is trend anomalies where LLM-STARS has very close performances with all other setups. This indicates that the trend anomaly is easier to detect than other anomaly types. The most striking improvement comes from incorporating relationship information into the modeling process, regardless of how these relationships are identified. This is evidenced by the significant performance gap between relationship-aware methods and the univariate baseline. Although simply finding relationships from time series similarities can achieve much better performance than univariate time series modeling, the performance improvement is smaller than approaches using semantic accounting use case relationships. Pairwise similarities cannot efficiently improve time series analysis when multiple regressors are present. Moreover, the similarity-based approach lacks the interpretability offered by our LLM-based method, which provides meaningful accounting-domain explanations for identified relationships.

Table 2: Time-series forecasting performance on SubledgerVirtualEnv Dataset

method	sMAPE	mase_1	mase_7	mase_{30}	msis_1	msis_7	msis_{30}
ablation_no_neighbor	28.4163	0.2598	0.2783	0.1859	2.3962	2.5674	1.7144
ablation_no_recon	28.7364	0.2688	0.2880	0.1923	2.4369	2.6110	1.7435
ablation_no_standardization	31.5136	0.3328	0.3566	0.2381	2.9748	3.1873	2.1283
baseline_univariate	53.8155	1.3002	1.3931	0.9302	9.6294	10.3173	6.8894
baseline_ts_similarity	32.4568	0.6044	0.6476	0.4324	4.0327	4.3207	2.8852
Proposed LLM-STARS	**27.8275**	**0.2584**	**0.2769**	**0.1849**	**2.3841**	**2.5544**	**1.7057**

5.5 Ablation Analysis

Impact of Standardization. Among all ablations, removing event description standardization leads to the most significant performance degradation. This highlights the importance of maintaining consistent representations across financial events. The standardization process, which uses a single LLM call to generate unified representations, effectively bridges the variation in initial individual LLM outputs. Additionally, this step serves as an implicit form of neighborhood information aggregation, as the standardization prompt considers multiple events within an account simultaneously.

Impact of Neighborhood Information. The relatively modest improvement from explicit neighborhood information aggregation suggests that, in our experimental dataset, the attribute names and values already contain substantial information. The standardization step's implicit neighborhood consideration may also contribute to this effect. Nevertheless, the consistent, albeit small, performance improvements indicate that neighborhood information remains valuable for generating more comprehensive representations.

Impact of Reconciliation Relationships. While reconciliation relationships show positive but modest improvements, this may undersell their potential value. Our current implementation focuses on one-to-one reconciliation patterns, but real-world reconciliations often involve more complex relationships. For example, a single bank deposit use case might reconcile with multiple credit card receivable use cases across different business units. Future work extending the model to capture these hierarchical relationships could yield more substantial improvements.

6 CONCLUSION

In this paper, we presented a novel framework that leverages Large Language Models to enhance time series analysis of subledger data through relationship modeling. Our approach addresses the fundamental challenge of non-standardized financial event representations in complex accounting systems and explicit relationship modelling using LLM. The framework successfully standardizes diverse financial event representations while preserving their accounting significance, effectively identifies meaningful relationships between subledger use cases including both reconciliation pairs and similar pattern relationships, and achieves superior forecasting and anomaly detection performance compared to univariate approaches. Importantly, the framework provides interpretable results through natural language descriptions that align well with human expert annotations.

Looking ahead, several promising directions for future research emerge. First, while our current time series analysis approach focuses on simple short-term (daily) reconciliation patterns, extending the framework to capture longer-term relationships could provide valuable insights for complex accounting cycles and delayed settlements. This could be done by choosing other multivariate time-series analysis models. Second, the relationships identified in our current work could be enhanced by incorporating explicit accounting knowledge graphs. This would enable representation of hierarchical relationships between accounting concepts, integration of regulatory requirements and compliance rules, and more sophisticated reasoning about transaction flows and their implications.

REFERENCES

[1] Estee Y Cramer, Evan L Ray, Velma K Lopez, Johannes Bracher, Andrea Brennen, Alvaro J Castro Rivadeneira, Aaron Gerding, Tilmann Gneiting, Katie H House, Yuxin Huang, et al. 2022. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the United States. *Proceedings of the National Academy of Sciences* 119, 15 (2022), e2113561119.

[2] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130* (2024).

[3] Graham C Gibson, Nicholas G Reich, and Daniel Sheldon. 2023. Real-time mechanistic bayesian forecasts of covid-19 mortality. *The annals of applied statistics* 17, 3 (2023), 1801.

[4] Qing Huang, Marco Schreyer, Nilson Michiles, and Miklos Vasarhelyi. 2024. Connecting the dots: Graph neural networks for auditing accounting journal entries. *Available at SSRN 4847792* (2024).

[5] Ming Jin, Huan Yee Koh, Qingsong Wen, Daniele Zambon, Cesare Alippi, Geoffrey I Webb, Irwin King, and Shirui Pan. 2024. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[6] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. 2023. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728* (2023).

[7] Ming Jin, Qingsong Wen, Yuxuan Liang, Chaoli Zhang, Siqiao Xue, Xue Wang, James Zhang, Yi Wang, Haifeng Chen, Xiaoli Li, et al. 2023. Large models for time series and spatio-temporal data: A survey and outlook. *arXiv preprint arXiv:2310.10196* (2023).

[8] Ming Jin, Yifan Zhang, Wei Chen, Kexin Zhang, Yuxuan Liang, Bin Yang, Jindong Wang, Shirui Pan, and Qingsong Wen. 2024. Position: What can large language models tell us about time series analysis. In *Forty-first International Conference on Machine Learning*.

[9] Vamsi Krishna Kommineni, Birgitta König-Ries, and Sheeba Samuel. 2024. From human experts to machines: An LLM supported approach to ontology and knowledge graph construction. *arXiv preprint arXiv:2403.08345* (2024).

[10] Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu. 2021. Revisiting time series outlier detection: Definitions and benchmarks. In *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 1)*.

[11] Peiwen Li, Xin Wang, Zeyang Zhang, Yuan Meng, Fang Shen, Yue Li, Jialong Wang, Yang Li, and Wenwu Zhu. 2024. RealTCD: temporal causal discovery from interventional data with large language model. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 4669–4677.

[12] Chen Liu, Shibo He, Qihang Zhou, Shizhong Li, and Wenchao Meng. 2024. Large language model guided knowledge distillation for time series anomaly detection. *arXiv preprint arXiv:2401.15123* (2024).

[13] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2020. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting* 36, 1 (2020), 54–74.

[14] Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. 2020. Dynotears: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*. Pmlr, 1595–1605.

[15] Sean J Taylor and Benjamin Letham. 2018. Forecasting at scale. *The American Statistician* 72, 1 (2018), 37–45.

[16] Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. 2024. Timexer: Empowering transformers for time series forecasting with exogenous variables. *arXiv preprint arXiv:2402.19072* (2024).

[17] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682* (2022).

[18] Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. 2024. Large language models for generative information extraction: A survey. *Frontiers of Computer Science* 18, 6 (2024), 186357.

[19] Daksha Yadav, Sabrina Zhang, and Tom Jin. 2023. Transformer based anomaly detection on multivariate time series subledger data. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*. https://www.amazon.science/publications/transformer-based-anomaly-detection-on-multivariate-time-series-subledger-data

[20] Wenzhuo Yang, Kun Zhang, and Steven CH Hoi. 2022. A causal approach to detecting multivariate time-series anomalies and root causes. *arXiv preprint arXiv:2206.15033* (2022).

[21] Shengbin Yue, Wei Chen, Siyuan Wang, Bingxuan Li, Chenchen Shen, Shujun Liu, Yuxuan Zhou, Yao Xiao, Song Yun, Xuanjing Huang, et al. 2023. Disc-lawllm: Fine-tuning large language models for intelligent legal services. *arXiv preprint arXiv:2309.11325* (2023).

[22] Yunhao Zhang and Junchi Yan. 2023. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*.

A PROMPTING

A.1 Label Space Generation

Due to prompt input size limitations, it is impractical to process all event descriptions simultaneously. We therefore employ a divide-and-conquer strategy to generate the label space efficiently:

(1) First, we group financial events by their associated account numbers, as events related to the same account typically serve similar business purposes
(2) Within each account number, we randomly batch the event descriptions and use LLM to generate labels for each batch
(3) A final consolidation step merges these account-specific labels to create the complete label space

The label generation process is conceptually similar to a clustering problem, where we can control the granularity through careful prompt engineering. In our implementation, we specifically design prompts to maintain meaningful distinctions between financial events. For example, we include instructions like "Labels should be distinguishable to recognize each key event in a life cycle. Do not group activities that might clear/reconcile each other." This ensures that the resulting labels capture functionally distinct activities, particularly those that may be involved in reconciliation relationships.

A.2 Prompt Examples

A.2.1 Node Description. Template used to generate text description for financial event nodes and accounting nodes

> Given a accounting segment/financial event defined by:
> * Key value mappings: {ATTRIBUTE KEY VALUE MAPPING}
> * Chart Of Account: {COA VALUE DESCRIPTIONS} (This only exist when describing accounting segment nodes)
> And its associated financial events/accounting segment summary:
> {A TABLE WITH EDGES AND NEIGHBORS INFO}
> As an accounting expert, you need to document this ledger segment/financial event from the following aspects:
> 1. The meaning of the segment/event
> 2. The segment/event's core function in the accounting workflow
> 3. Financial events that manages this segment. / Accounting segments that are managed by this event.
> Expected response format: <Description> [Structured technical analysis of the segment] </Description>
> **Notes**: {SOME ADDITIONAL NOTES} (For example, keep in mind the data in the table may not be complete so do not entirely rely on the numbers.)

A.2.2 Label Space Generation. Template used to merge financial event node and generate a labelling space:

> {EVENT LIST}
> The above are {EVENT NUMBER} summaries for financial events which trigger ledger booking. As an accountant, you are standardizing those event (Analyze the events and merge similar ones) and generate labels for them. Each label should be accompanied by a description to explain the detailed meaning of the event such as the functional purpose of it and the role it plays in accounting cycle.
> Your output should be in the following format:
> <Events> <Event> <Label> [Event label you choose] </Label> <Description> [Description of the event] </Description> </Event> </Events>
> **Notes**: {SOME ADDITIONAL NOTES}

A.2.3 Label Assignment. Prompt to assign one label to each financial event

> {EVENT TO LABEL}
> {POSSIBLE LABELS}
> The above is an Event Description and all possible event labels that the event could belong to. The event is related to an account: {ACCOUNT DESCRIPTION}. Event descriptions are summaries of the events provided by accounting experts.
> As an accounting expert, you are going to label each event. You have done the first step to list all of the labels and create a detailed explanation for it (see PossibleLabels above). Now you need to assign one label to each event.
> Output the result in the following format <Label></Label>
> **Notes**: {SOME ADDITIONAL NOTES}

A.2.4 Possible Recon Events Description. Prompt to extract descriptions of events that may clear/reconcile with the target event:

> **Subledger Account Info**: {ACCOUNT INFO}
> **Financial Event Info**: {EVENT INFO}
> The above label and description pair represents a financial event that books to a subledger account (see account info above). As an accounting expert, you are doing reconciliation for this event. The first step is to write a description to represent the possible events IN THE SAME ACCOUNT that clears or reconciles this event.
> Output in the following format <Description></Description>
> **Notes**: {SOME ADDITIONAL NOTES}

A.3 Standardized Event Representation Example

> <Label> Customer Advance - Retail Performance Obligation Fulfillment </Label>
> <Description> This event occurs when *** fulfills its performance obligation to customers (by delivering goods or services) for which payment was previously received and recorded as a customer advance. The transaction decreases the *** liability account through negative transaction amounts (for liability reduction), effectively converting the advance payment liability into recognized revenue. This event represents the completion of the revenue recognition process where the previously recorded liability is cleared as *** satisfies its performance obligation to the customer. This typically occurs when orders that were previously paid for are shipped or delivered. </Description>

A.4 Possible Recon Events Description Example

> The reconciling events would be those that initially establish the customer advance liability in the *** account. These would include customer payment receipts or settlement confirmations from payment processors for goods or services that *** has not yet delivered or fulfilled. These initial events would create positive transaction amounts in this liability account, representing funds received in advance of *** completing its performance obligations. When these advance payments are recorded, they increase the liability balance, which is later reduced when the "Customer Advance - Retail Performance Obligation Fulfillment" event occurs. The full reconciliation cycle involves matching the initial customer advance receipts with their corresponding fulfillment events to ensure all customer advances are properly converted to revenue once performance obligations are satisfied.

B TIME SERIES FORECASTING METRICS DETAIL

$$\text{sMAPE} = \frac{200}{H} \sum_{h=1}^{H} \frac{|y_h - \hat{y}_h|}{|y_h + \hat{y}_h|}$$

$$\text{MASE} = \frac{1}{H} \sum_{h=1}^{H} \frac{|y_h - \hat{y}_h|}{\frac{1}{H-S} \sum_{j=S+1}^{H} |y_j - \hat{y}_{j-s}|}$$

$$\text{SIS} = \frac{1}{H} \frac{\sum_{h=1}^{H} (u_h - l_h) + \frac{2}{\alpha}(l_h - y_h) I y_h < l_h + \frac{2}{\alpha}(y_h - u_h) I y_h > u_h}{\frac{1}{H-S} \sum_{j=S+1}^{H} |y_j - y_{j-S}|} \tag{9}$$

Where h is the forecast horizon, from 1 to H; y_h is the actual value at horizon h; $\hat{y}_h$ is the predicted value at horizon h; u_h and l_h are the upper and lower bounds of the prediction interval at horizon h; S is the seasonal period; α is the significance level for the prediction interval ($\alpha = 0.05$ in our experiments.).