Vol. 8, No. 2, April 2026

Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

Sequential Requirements Prediction in Synthetic Fintech-Like Backlogs Using an Interpretable Hybrid of Transition Rules and Transformer Models

Diana Laily Fithri[1], Soni Adiyono[2], Muhammad Arifin[3]

[1,2,3]Information System Department, Faculty of Engineering, Muria Kudus University, Kudus, Indonesia

Received:
September 10, 2025
Revised:
March 10, 2026
Accepted:
April 1, 2026
Published:
April 12, 2026

Corresponding Author:
Author Name*:
Diana Laily Fithri
Email*:
diana.laily@umk.ac.id

DOI:
10.63158/journalisi.v8i2.1469

Abstract. Fintech software development is characterized by rapid product iteration and stringent regulatory requirements, resulting in changing requirements as an interrelated set rather than individual elements. In this research, Sequential Requirements Prediction is proposed as a decision support task in Requirements Engineering, where the time-ordered prefix of completed backlog items is used to predict the next likely canonical requirement type as Top-k ranked output. To mitigate noise and inconsistency in backlog data, an LLM-aided semantic normalization step maps diverse requirement descriptions to a closed set of fintech requirement types. The research compares an interpretable rule-based Markov-1 predictor with Transformer-based sequential predictors under a case-level time-aware split. The proposed method is evaluated on a synthetic fintech-like backlog dataset consisting of 900 cases, 5,252 events, and 18 canonical requirement types. The best-performing model, Transformer + normalization + augmentation (M4), achieved Recall@5 = 0.638889, MRR@5 = 0.536806, and NDCG@5 = 0.566667. These results surpassed the rule-based predictor and non-normalized Transformer model. In addition, augmentation further improved Recall@5 from 0.493056 to 0.527778 in the rare-type subset. These findings suggest the methodological promise of the proposed framework for sequence-aware and compliance-conscious backlog analytics in synthetic fintech-like settings.

Keywords: Requirements Engineering, Sequential Requirements Prediction, Synthetic Fintech-Like Backlogs, Semantic Normalization, Transformer

1. INTRODUCTION

Requirements Engineering (RE) remains the primary mechanism for translating stakeholder intent into buildable software increments, yet it is increasingly executed under conditions of volatility: shifting customer expectations, evolving regulatory constraints, and continuous delivery cadences. In fintech, these pressures are amplified because product changes are tightly coupled with compliance requirements and risk controls, while competition encourages rapid experimentation. In practice, teams therefore manage requirements as living backlogs (e.g., user stories, epics, change requests, issue tickets) whose content and priority evolve over time. This creates a recurrent decision problem: given the current backlog state and recent delivery history, what should be specified next (or what is likely to be requested next), and how can that prediction be made reliably enough to support planning, prioritization, and early risk analysis?

A promising way to formalize this problem is to treat backlog evolution as a sequence modeling task. "Sequential Requirements Prediction" can be framed as predicting the next requirement (or requirement category, dependency, or refinement) conditioned on an ordered history of prior requirements and contextual signals (e.g., release cycles, incident patterns, regulatory changes, customer segments) [1]. This framing is technically analogous to sequential recommendation, where models learn order-dependent preferences and transitions from interaction histories. Recent surveys in sequential recommendation highlight how modern approaches especially Transformer-style architectures and data-centric improvements capture temporal dependencies and long-range patterns that simpler Markovian or bag-of-events baselines miss [2] [3]. For RE, the implication is direct: if backlog items exhibit recurring trajectories (e.g., "KYC enhancement → fraud-rule tuning → dispute workflow change"), then learning these trajectories can support early identification of likely upcoming requirements and reveal latent structure in the requirements stream [4] [5].

At the same time, requirements artifacts are text-heavy and semantically nuanced, which makes purely ID-based sequence learning brittle across teams and domains. This is where Large Language Models (LLMs) become strategically relevant. A recent systematic literature review of LLMs in RE (covering studies from 2023–2024) shows rapid growth

of LLM-enabled RE automation, but also a concentration of work in elicitation and validation rather than downstream decision-support tasks such as prioritization and management analytics [6]. Complementary guidance has emerged on how to select and adapt LLMs for NLP-intensive RE tasks, emphasizing systematic workflows and the need to align model choice, prompting, and evaluation with the RE objective [7]. Meanwhile, road mapping work at the intersection of formal RE and LLMs underscores that any operational use must address correctness and trustworthiness either by constraining the model with formal methods or by making formal reasoning more accessible through LLM interfaces [8] [9]. Together, these strands suggest that LLMs are mature enough to be useful for requirements representation and augmentation, but they must be integrated with rigorous evaluation and governance to be credible in fintech contexts.

Empirical results from adjacent software engineering artifacts reinforce both the opportunity and the caution. For instance, ChatGPT has been evaluated for user story quality assessment and found to align reasonably with human judgments, while still exhibiting stability and trustworthiness issues that matter for non-expert usage [10]. Similarly, in bug repositories another class of natural-language-heavy engineering artifacts LLM-based hybrid methods have improved duplicate bug report detection by injecting semantic understanding where traditional bag-of-words approaches struggle [6]. Yet benchmarks such as SWE-bench demonstrate that end-to-end automated issue resolution remains difficult for language models, even when the task is grounded in real repositories and pull requests [11]. These findings collectively motivate a design stance for Sequential Requirements Prediction: use LLMs where they are strongest (semantic encoding, controlled generation, data enrichment), and pair them with robust sequence learners and evaluation protocols rather than expecting unstructured prompting to produce dependable planning outputs [12] [13].

A key technical bottleneck for learning-based RE is data scarcity and label sparsity, especially when the prediction target is specific (e.g., next requirement type, next dependency link, next compliance-driven change). Recent work in requirements traceability shows that LLM-driven data augmentation can mitigate scarcity by generating plausible additional trace links and enriching training sets, yielding substantial performance gains when paired with an improved encoder architecture [11]. More broadly, surveys of LLM-based data augmentation outline emerging paradigms for controllable

synthetic data generation and its use for downstream training, while also emphasizing open challenges (e.g., distribution shift, contamination, evaluation leakage) that must be managed carefully [7] [14]. For Sequential Requirements Prediction in fintech, these insights point toward a concrete research direction: construct sequence datasets from issue trackers/backlogs; represent items using LLM-derived semantic features; and apply principled augmentation (e.g., paraphrases, controlled rewrites, scenario-conditioned variants) to strengthen generalization without undermining validity.

In terms of novelty, this research contributes to the recent line of LLM-based Requirements Engineering research by moving from a collection of isolated NLP tasks to a more temporally grounded decision support task: next requirement type prediction in a fintech-like backlog stream. This research has three main novelty aspects. First, it provides research setting for Sequential Requirements Prediction, where prediction targets and Top-k evaluation are well-defined. Second, it proposes a semantic normalization stage that relies on LLMs to transform heterogeneous text in the backlog stream into a standard requirement type, thus addressing issues of noise and sparsity in the data. Third, it presents a combination of interpretable rule-based transition modeling and Transformer-based sequence prediction, including augmentation for improved Top-k ranking in sparse conditions. These novelty aspects are validated in research setting with a synthetically generated fintech-like backlog stream, so it is expected that the study's findings be understood as a proof of concept of methodological potential, rather than a direct validation of industrial applicability.

2. METHODS

Figure 1 presents the conceptual framework of the proposed study. It summarizes how fintech backlog artifacts are transformed through LLM-assisted semantic normalization into canonical requirement types, which are then processed by sequential prediction models and evaluated under a time-aware recommendation setting.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

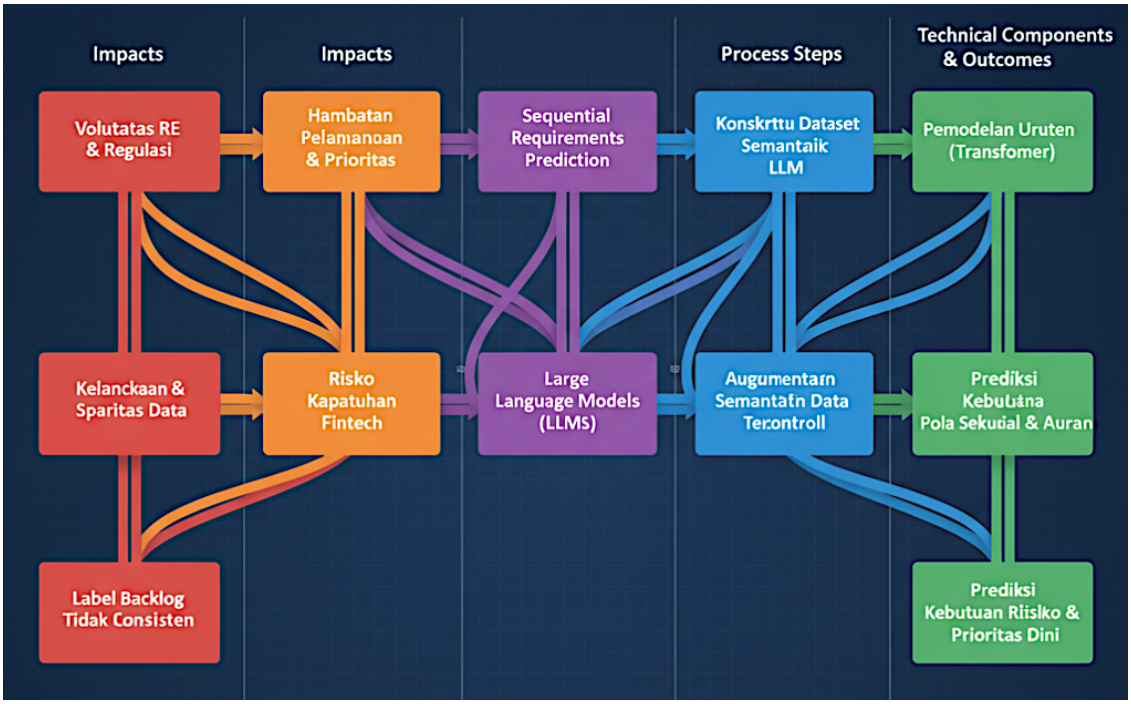

Figure 1. Conceptual Framework of LLM-Augmented Sequential Requirements Prediction in Fintech

2.1. Proposed Framework and Research Workflow

This study proposes a hybrid framework for Sequential Requirements Prediction that combines LLM-assisted semantic normalization, interpretable transition-rule modeling, Transformer-based sequence learning, controlled augmentation, and time-aware Top-k evaluation. This research frames Sequential Requirements Prediction as a time-ordered decision-support problem: given a partial history of implemented backlog items within a coherent development stream (the *case*), the goal is to predict the next requirement type (or produce a Top-*k* ranked recommendation list). The pipeline is designed to be reproducible and to reflect contemporary guidance on using LLMs in RE: the LLM is treated as a controlled component in the workflow (with explicit inputs/outputs and logging) [15].

The pipeline has four stages: (1) construct sequences from issue-tracker/backlog history with strict temporal ordering; (2) apply semantic normalization to map heterogeneous user stories into a closed taxonomy of canonical requirement types; (3) train sequential predictors on normalized sequences; and (4) evaluate on temporally held-out data using recommendation-style metrics. The motivation is aligned with recent RE literature

emphasizing the rapid growth of LLM-enabled RE methods and the need for systematic evaluation and trustworthiness considerations [16]. Figure 2 illustrates the proposed end-to-end workflow, starting from backlog-event sequence construction, followed by LLM-assisted semantic normalization, optional augmentation on the training split, sequential model learning, and final Top-k evaluation under a case-level time-aware split.

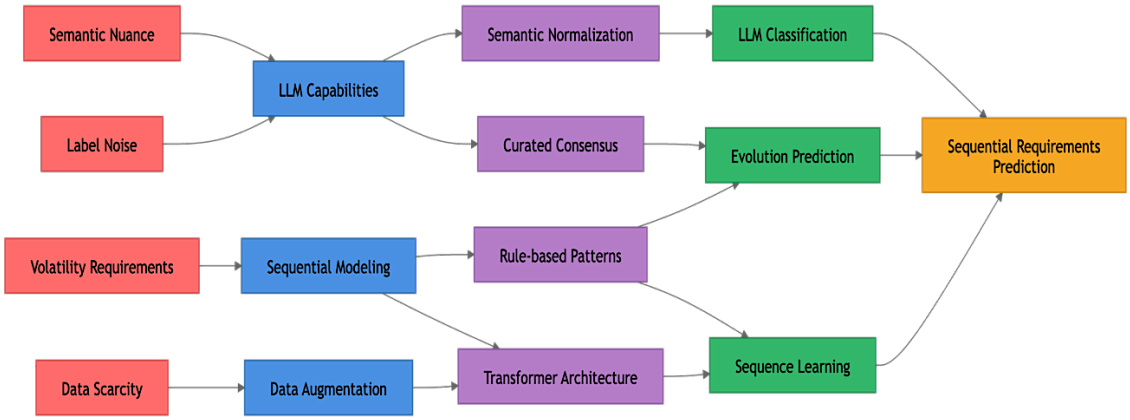

Figure 2. Sequential Requirements Prediction Flow: LLM-Assisted Normalization, Controlled Augmentation, and Sequence Learning.

Some parts of this pipeline follow best practices for sequence prediction and recommendation systems, such as using a Markov-1 baseline for local interpretable transition analysis, a sequence learner based on a Transformer architecture for prefix-based predictions, a set of Top-k ranking metrics for shortlist-oriented evaluation, and a case-level time-aware split to avoid temporal leakage. The main contribution of this manuscript in terms of methodology is not in any of these components individually but rather in their combination for a novel application of sequential requirement prediction in synthetic fintech-like backlogs. The contribution of this manuscript can be characterized by three parts: (i) an LLM-assisted semantic normalization step to transform heterogeneous backlog items into a canonical form of requirement types, (ii) a hybrid predictive approach combining interpretable transition analysis and learned sequence modeling, and (iii) a controlled data augmentation approach, applied only to the training set for ranking improvement of rare requirement types.

2.2. Synthetic Fintech-Like Dataset Construction and Case Definition

Data source and unit of analysis. The primary artifacts are backlog items (user stories/issues/feature tickets) including textual fields (title/description) and lifecycle metadata (timestamps, component/labels, links). Each event is defined as an item reaching a stable state (e.g., "Done/Accepted"), and each case represents a coherent stream such as an Epic/Feature, a product component over a release window, or a customer journey slice (e.g., KYC upgrade). Events are ordered by acceptance timestamps to form sequences, and items with ambiguous/missing timestamps are removed to preserve temporal integrity. Because access to real industrial multi-project fintech backlogs is restricted, this study evaluates the proposed framework on a synthetic fintech-like dataset designed to approximate realistic backlog trajectories. The dataset was constructed by defining 18 canonical fintech requirement types, generating 900 case sequences with temporally ordered events, and curating transition patterns that reflect plausible fintech development flows such as KYC verification, AML screening, audit logging, notifications, dispute handling, and risk-control follow-ups.

Canonical taxonomy for fintech requirements. A closed set of canonical requirement types is defined (e.g., KYC_VERIFICATION, AML_SCREENING, LIMIT_CHANGE, DISPUTE_FLOW, RISK_RULE_TUNING, PAYMENT_ROUTING). This reduces label noise and enables learning over stable tokens rather than inconsistent free-text labels.

LLM-assisted semantic normalization. Each backlog item is mapped to one canonical type using an LLM-driven classification prompt that ingests consolidated text (title + description + selected metadata) and outputs: (i) the canonical type and (ii) a short rationale (stored for audit). To improve stability, we use repeated runs and majority voting ("best-of-n"), consistent with prior findings that output stability can be improved via multi-sampling for requirements-adjacent judgments [17]. This design follows systematic guideline-oriented recommendations for LLM use in RE (controlled inputs, constrained output space, and explicit evaluation) [18] [19].

2.3. LLM-Assisted Semantic Normalization

Models. We evaluate two complementary predictors to balance performance and interpretability for RE decision support:

1) Interpretable rule-based sequential predictor. Frequent sequential transitions between canonical tokens are extracted and converted into next-type recommendation rules (ranked by association strength). This baseline provides transparent "if-prefix-then-next" explanations suitable for RE governance.
2) Learned sequential predictor (Transformer-style). A sequence model predicts the next canonical token from the observed prefix, optionally incorporating side information (e.g., component/priority embeddings). The model choice is justified by recent sequential recommendation surveys describing the effectiveness of attention-based architectures for order-dependent prediction tasks [20].

Optional enhancement: LLM-based augmentation (training only). To mitigate sparsity, we optionally augment training data with controlled LLM-generated paraphrases and semantically equivalent variants that preserve the canonical label. This is motivated by recent work on LLM-based data augmentation (including constraints and pitfalls such as distribution shift and evaluation leakage) [14]. We also ground the augmentation rationale in evidence that LLM augmentation can improve requirements-related learning tasks under limited labeled data (shown, for example, in traceability augmentation studies) [21].Temporal split and metrics. We use a time-aware split (train on earlier periods, test on the most recent period) to avoid leakage, consistent with best practice in sequence prediction settings. Primary metrics are HitRate/Recall@k, MRR, and NDCG@k, reflecting the practical objective of presenting a small ranked shortlist for planning rather than a single hard label. Coverage (fraction of prefixes for which recommendations are produced) is reported as an operational metric [22] [23].

2.4. Sequential Models and Controlled Augmentation

To test the proposed framework, this research compares four key predictive configurations for balancing interpretability, ranking quality, and robustness in sparse data settings.

1) M1: Rule-based Markov-1 predictor.

The first model is an interpretable sequential baseline based on first-order transitions between canonical types of requirements. Given a prefix, M1 recommends types of requirements that tend to appear most frequently after the last token in the training

data. The ranking is based on transition strength, derived from observed sequential data. The model is intentionally simple and interpretable, making it suitable as a governance-oriented baseline in Requirements Engineering settings where "if-then" explainability is critical.

2) M2: Transformer without semantic normalization.

The second model is a learned sequential predictor, where the entire prefix is used to predict the next requirement type. For this model, M2, the requirement labels are used in their original form before applying semantic normalization. This model is used to evaluate whether a Transformer architecture can learn to predict the next requirement type given a sequence of requirement types, including a comparison of its performance when faced with noisy and inconsistent labels.

3) M3: Transformer with semantic normalization.

The third model is similar to M2, except that it uses normalized requirement types, where each requirement type is normalized to one of the canonical requirement types by applying the LLM-based semantic normalization. This model is used to evaluate the main hypothesis that by normalizing requirement types, the overall noise and sparsity of requirement labels in a backlog are reduced, thus making it easier for a sequence predictor to learn.

4) M4: Training-only augmentation protocol.

A strict leakage-resistant protocol is employed, where the training set is the only set of data on which the augmentation process is applied. This ensures that the test and validation sets remain unaltered and that the improvement in the model is not due to the leakage of information between the training and test set. In this case, the augmented data is provided with the original canonical label and is only employed for the improvement of the model's representation of the less observed patterns. This is particularly significant in the context of the study because a time-aware protocol is employed.

2.5. Time-Aware Evaluation Protocol and Metrics

The proposed framework is evaluated under a case-level time-aware split to preserve temporal validity and prevent information leakage. Rather than randomly splitting

instances, cases are assigned to training, validation, and test partitions according to the timestamp of their last event, so that earlier cases are used for training and the most recent cases are reserved for evaluation. In the present dataset, this protocol yields 688 training cases, 76 validation cases, and 136 test cases.

To operationalize next-requirement prediction, each case sequence is converted into prefix–next pairs. For a sequence of length T, all prefixes from length 1 to $T-1$are used to predict the immediately following requirement type. This produces 3,335 training instances, 367 validation instances, and 650 test instances. Because the practical objective is to support backlog refinement with a ranked shortlist rather than a single deterministic output, evaluation is conducted using Top-k recommendation metrics. Let $\hat{Y}_k(x_i)$denote an ordered list of the top-k predicted requirement types for prefix x_i, and let $y_i$be the true next type. We report Recall@k, MRR@k, NDCG@k, and Coverage@k. Recall@k (HitRate@k), For single-label next-item prediction, Recall@k equals HitRate@k:

$$\text{Recall@k} = \frac{1}{N}\sum_{i=1}^{N} 1\left(y_i \in \hat{Y}_k(x_i)\right) \tag{1}$$

where $1(\cdot)$is an indicator function that equals 1 if the condition is true and 0 otherwise. Mean Reciprocal Rank (MRR@k), MRR@k measures how high the correct next requirement is ranked within the top-k list. Let rank_ibe the rank position of $y_i$in $\hat{Y}_i^{(k)}$. If $y_i$does not appear in the top-k, its contribution is 0. Then:

$$\text{MRR@k} = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}\frac{1}{\text{rank}_i} & \text{if } y_i \in \hat{Y}_k(x_i) \\ 0 & \text{otherwise}\end{cases} \tag{2}$$

MRR is rank-sensitive and directly reflects how quickly a correct recommendation appears in a shortlist.

Normalized Discounted Cumulative Gain (NDCG@k), NDCG@k is used as a rank-sensitive metric that gives higher weight to correct predictions appearing earlier in the ranked list. For binary relevance:

$$\mathrm{DCG@k}_i = \sum_{r=1}^{k} \frac{\mathrm{rel}_{i,r}}{\log_2(r+1)} \tag{3}$$

where $rel_r = 1$if the item at rank $r$is the true next requirement type, and $rel_r = 0$otherwise. NDCG is then computed as:

$$\mathrm{NDCG@k} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{DCG@k}_i \tag{4}$$

Because each instance has only one relevant next item, $\mathrm{IDCG@}k = 1$when that item is ranked first. Coverage@k. Coverage measures the fraction of instances where the model produces a non-empty recommendation list:

$$\mathrm{Coverage@k} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left(|\ \hat{Y}_k(x_i)\ | > 0\right) \tag{5}$$

This metric is operationally important because some interpretable rule-based methods may fail to produce recommendations for prefixes whose outgoing transitions were not observed during training. Overall, this evaluation protocol is designed to reflect realistic deployment conditions in which requirement histories are observed over time, future instances must remain unseen during training, and practical utility depends on the quality of a short-ranked recommendation list rather than on single-label classification accuracy alone.

3. RESULTS AND DISCUSSION

3.1. Dataset Overview and Experimental Setup

The evaluation dataset consists of sequential requirement events derived from backlog artifacts, where each event corresponds to a completed/accepted backlog item mapped into a canonical requirement type. The synthetic dataset contains 900 cases, 5,252 events, and 18 canonical requirement types. A case-level time-aware split is used to prevent temporal leakage: cases are assigned to training, validation, or test partitions based on

the timestamp of their last event. This yields 688 training cases, 76 validation cases, and 136 test cases.

To operationalize next-requirement prediction, each case sequence $S = (e_1, e_2, \dots, e_T)$is converted into prefix–next pairs $(S_{1:t}, e_{t+1})$for all $t \in [1, T-1]$. The resulting sample sizes are 3,335 (train), 367 (validation), and 650 (test). The average prefix length is short (approximately three events), reflecting the practical reality that many fintech delivery trajectories (e.g., KYC/AML follow-ups, audit logging, notification) form compact chains. These characteristics create a challenging yet realistic prediction setting: frequent cross-cutting requirement types dominate the head of the distribution, while specialized features appear in the long tail. Consequently, we evaluate performance using Top-k recommendation metrics rather than strict next-1 accuracy alone.

3.2. Performance Evaluation

We evaluate five models to balance predictive quality and interpretability while isolating the effects of the proposed novelty components. M0 (Popularity) serves as a lower bound by recommending globally frequent requirement types; it benefits from recurring cross-cutting items (e.g., audit/notification) but cannot represent dependency structure, so it is weak when the "next" requirement is context-dependent.

M1 (Rule-based Markov-1) improves substantially by modeling local transitions: it recommends what most often follows the last observed requirement type. This baseline is especially strong in fintech journeys with repetitive short-range patterns (e.g., KYC → AML, functional change → audit/logging) and is attractive operationally because its recommendations are directly explainable and auditable. However, Markov-1 is limited at branching points where longer history matters.

To handle richer context, M2 (Transformer without normalization) learns from the full prefix, but performance is constrained by noisy and inconsistent labels that fragment semantically equivalent requirements. M3 (Transformer + normalization) addresses this by applying LLM-assisted semantic normalization into canonical requirement types (Novelty N1), reducing sparsity and improving ranking quality in Top-k recommendations. M4 (Transformer + normalization + augmentation) further adds controlled augmentation

(Novelty N3), yielding its largest benefit under sparsity particularly for rare requirement types and short prefixes rather than uniformly boosting all cases.

To assess the effectiveness of the proposed framework, the study first compares all models under the overall test setting using Top-k recommendation metrics. This comparison is intended to show how performance changes from a frequency-based baseline, to an interpretable rule-based sequential model, and finally to Transformer-based models with semantic normalization and controlled augmentation. The metrics reported in Table 3 capture both recommendation accuracy and ranking quality, allowing a more complete view of how well each model supports next-requirement prediction in fintech-like backlog sequences.

Table 3. Overall test metrics

Model	Recall@1	Recall@3	Recall@5	Recall@10	MRR@5	NDCG@5	Coverage@5
M0 Popularity	0.263889	0.430556	0.500000	0.590278	0.355556	0.406250	1.000.000
M1 Rule-based	0.402778	0.541667	0.583333	0.638889	0.481944	0.518750	0.673611
M2 Transformer (no norm)	0.437500	0.569444	0.604167	0.652778	0.508333	0.538889	1.000.000
M3 Transformer + norm	0.472222	0.597222	0.631944	0.666667	0.531250	0.561111	1.000.000
M4 Transformer + norm + aug	0.479167	0.604167	0.638889	0.670139	0.536806	0.566667	1.000.000

Table 3 shows that the popularity baseline (M0) provides a useful lower bound, but it is insufficient for modeling journey-level dependencies in fintech backlog sequences. Although M0 achieves a non-trivial Recall@5 of 0.500000, its relatively weak Recall@1 (0.263889) and lower rank-sensitive scores indicate that global frequency alone cannot adequately capture what requirement type is likely to occur next. In contrast, the rule-based Markov-1 model (M1) substantially improves performance, reaching Recall@5 = 0.583333, MRR@5 = 0.481944, and NDCG@5 = 0.518750, which suggests that fintech backlog evolution contains meaningful local transition regularities. However, M1 achieves Coverage@5 = 0.673611, indicating that the rule-based approach cannot always produce

recommendations when the last observed token has insufficient outgoing transitions in the training data.

The Transformer-based models (M2–M4) further improve ranking quality by modeling the full prefix rather than relying only on the most recent requirement type. Among them, M4 achieves the best overall performance, with Recall@5 = 0.638889, MRR@5 = 0.536806, and NDCG@5 = 0.566667. These results indicate that learned sequence models are better suited to ambiguous backlog states, where several next requirements are plausible and broader context is needed to rank the correct candidate near the top of the recommendation list. From a Requirements Engineering perspective, this is operationally important because higher MRR and NDCG imply that analysts can work with a shorter and more precise shortlist during backlog refinement and planning.

3.3. Effect of Semantic Normalization

To isolate the contribution of semantic normalization, we compare M2 (Transformer without normalization) and M3 (Transformer with canonicalized requirement types). The comparison shows consistent improvement across all main ranking metrics. In particular, Recall@5 increases from 0.604167 to 0.631944, MRR@5 from 0.508333 to 0.531250, and NDCG@5 from 0.538889 to 0.561111. These results support the hypothesis that semantic normalization reduces label fragmentation and sparsity. By mapping heterogeneous backlog expressions into a stable set of canonical requirement types, the sequential model can learn more reliable transition patterns. From a Requirements Engineering perspective, this suggests that semantic normalization is not merely a preprocessing convenience, but a central representational step that improves both learnability and interpretability in backlog analytics.

3.4. Effect of Controlled Augmentation on Rare Types

While the overall test results provide a general view of model effectiveness, they do not fully reveal how the models behave on infrequent requirement categories. In practice, rare requirement types are especially important because they often correspond to specialized compliance, risk-control, or exception-handling scenarios that are underrepresented in historical data. For this reason, a separate analysis is conducted on the rare-type subset to examine whether semantic normalization and controlled

augmentation provide additional benefit under sparse conditions. The corresponding results are summarized in Table 4.

Table 4. Rare-type subset performance (synthetic; bottom 30% frequency)

Model	Recall@5 (rare)	MRR@5 (rare)	NDCG@5 (rare)
M1 Rule-based	0.430556	0.351389	0.369444
M3 Transformer + norm	0.493056	0.392361	0.415278
M4 Transformer + norm + aug	0.527778	0.411111	0.431250

Table 4 highlights the effect of controlled augmentation in sparse prediction settings, specifically for the rare-type subset. The results show that M4 outperforms both the interpretable rule-based baseline (M1) and the normalized Transformer without augmentation (M3). Compared with M3, augmentation improves Recall@5 from 0.493056 to 0.527778, MRR@5 from 0.392361 to 0.411111, and NDCG@5 from 0.415278 to 0.431250. This pattern suggests that controlled augmentation is most beneficial when empirical support for certain requirement types is limited, because it strengthens the learning signal for underrepresented sequential patterns without altering the canonical label space. At the same time, the gain observed in the rare-type subset is more pronounced than the gain in the overall test set, indicating that augmentation should be interpreted as a targeted enhancement for sparse conditions rather than as a universally dominant improvement strategy. In practical terms, this means that augmentation is especially useful for long-tail requirement categories and short contexts, where purely empirical sequence evidence is insufficient. Because the evaluation is conducted on a synthetic fintech-like dataset, these results should be interpreted as evidence of methodological promise rather than as full real-world deployment validation.

3.5. Discussion

The results yield several key observations. First, the stronger results for the rule-based Markov-1 model imply that fintech backlogs do have stable short-range transitions. Thus, interpretable transition rules are an important baseline, particularly in governance-sensitive contexts. Second, the stronger MRR@5 and NDCG@5 scores for the Transformer-based models indicate that learned sequence models perform better when multiple possible next requirements are plausible. This is particularly important after checkpoint-like requirement states such as AML screening or risk scoring, in which the

next requirement may branch out in several possible and legitimate ways. Third, the consistent improvement over the range from M2 to M3 demonstrates that semantic normalization is indeed one of the most important contributions of the proposed framework. The reduction of lexical inconsistency and label fragmentation is seen to significantly improve the quality of the sequential learning and the stability of the backlog evolution. Lastly, the rare-type analysis demonstrates that augmentation is most beneficial when it is controlled in long-tail contexts in which there are few empirical examples.

The above findings can also be said to be in line with other previous works in neighboring domains of software engineering and requirements-oriented analytics. Research on sequential recommendation has demonstrated that, in general, attention-based sequence models perform better than local transition models, particularly in cases where prediction is influenced by more complex context, such as in scenarios where there is a need to rank multiple possible next actions, as opposed to making a single determinate prediction. Similarly, recent Requirements Engineering and traceability research have highlighted the importance of representation quality and data preparation on learning performance, particularly in cases where there is a significant amount of text in the artifacts, which may be semantically inconsistent or sparse. The above findings can be said to be in line with the above-mentioned research, as normalization improves ranking quality by making requirement type representation more stable, whereas augmentation is most beneficial in cases with rare types, where there is limited support.

Practically speaking, it appears that there is a distinct boundary beyond which settings benefit from the Transformer-based modeling and those settings in which it is less valuable. In settings in which backlog trajectory is relatively short, transitions are relatively local and stable, and in which the primary operating constraint is either transparency or auditability, it is likely that the Markov-1 baseline will be adequate to provide useful next-step recommendations with low complexity. In settings in which requirement evolution is subject to longer prefix context, branching checkpoints, or in which there are multiple competing candidates for the next-step requirements, such as in response to compliance review, AML checks, or risk control, it is in these settings in which Transformer-based models are more valuable in ranking the plausible candidates and generating a useful shortlist.

At the same time, however, these results should be viewed with caution. The evaluation was carried out on a synthetic fintech-like dataset and not on a live industrial multi-project backlog repository. As such, the results should be seen as indicative of the potential of the methodology and its internal consistency, but not yet as evidence of the effectiveness of the deployment.

To better understand the operational limits of the proposed framework, we examine several typical error patterns. A first challenge arises at branching checkpoints. After requirements such as AML screening or risk scoring, the subsequent requirement may diverge into multiple alternatives, such as dispute hardening, fraud-rule adjustment, or limit-control refinement. When only canonical tokens are modeled, these branches may appear ambiguous. A second issue is the over-recommendation of cross-cutting requirements such as audit logging and notifications. Because these requirement types are highly frequent across many cases, they can dominate the recommendation shortlist and suppress more specific next-step candidates. A third challenge concerns long-tail underprediction. Rare requirement types remain difficult to recommend reliably without either richer contextual features or training-time augmentation. The rare-type results indicate that controlled augmentation partially reduces this problem, although it does not eliminate it completely. From a practical perspective, the proposed approach should be positioned as decision support rather than full automation. Its value lies in generating ranked candidate requirements that may accelerate backlog refinement, surface likely compliance-related follow-ups earlier, and document recurring requirement trajectories for process learning and standardization.

Several limitations should be acknowledged. First, construct validity depends on the granularity of the canonical taxonomy. A taxonomy that is too coarse may artificially inflate predictive performance, whereas a taxonomy that is too fine may increase sparsity and reduce learnability. Second, internal validity depends on the strict enforcement of the case-level time-aware split. Any temporal leakage would artificially increase performance and weaken the credibility of the evaluation. Third, external validity remains limited because the present study uses a synthetic fintech-like dataset. Sequence patterns learned in this setting may not fully reflect the diversity of real organizational backlogs across teams, products, or release processes. Finally, the LLM-assisted components introduce their own risks. In real data settings, semantic normalization may

occasionally vary across runs or misclassify ambiguous items. For this reason, constrained output spaces, repeated runs, majority voting, and audit logging remain important safeguards for trustworthy deployment.

4. CONCLUSION

This study proposed a hybrid framework for Sequential Requirements Prediction in synthetic fintech-like backlogs by combining interpretable transition rules, Transformer-based sequence modeling, LLM-assisted semantic normalization, and controlled augmentation. The results show that sequence-aware models consistently outperform frequency-based recommendation, while the rule-based baseline remains competitive as an auditable reference in settings with stable short-range transitions. Semantic normalization improves ranking quality by reducing label fragmentation, and controlled augmentation provides additional benefit for rare requirement types under sparse conditions. Overall, the findings indicate the methodological promise of the proposed framework for sequence-aware backlog analytics and decision support. However, because the evaluation was conducted on a synthetic fintech-like dataset rather than on a live industrial multi-project repository, the results should be interpreted as proof-of-concept evidence rather than direct validation of deployment readiness. Future work should therefore validate the framework on real organizational backlog data, incorporate richer contextual features such as component, priority, release phase, and regulatory triggers, and evaluate human-in-the-loop usage to strengthen trustworthiness and practical usefulness in Requirements Engineering workflows.

REFERENCES

[1] S. V. A. Ammu, S. S. Sehra, S. K. Sehra, and J. Singh, "Amalgamation of Classical and Large Language Models for Duplicate Bug Detection: A Comparative Study," *Computers, Materials and Continua*, vol. 83, no. 1, pp. 435–453, 2025, doi: 10.32604/cmc.2025.057792.

[2] T. F. Boka, Z. Niu, and R. B. Neupane, "A survey of sequential recommendation systems: Techniques, evaluation, and future directions," *Inf. Syst.*, vol. 125, p. 102427, 2024, doi: 10.1016/j.is.2024.102427.

[3] L.-W. Pan, W.-K. Pan, M.-Y. Wei, H.-Z. Yin, and Z. Ming, "A survey on sequential recommendation," *Front. Comput. Sci.*, vol. 20, no. 3, p. 2003606, 2025, doi: 10.1007/s11704-025-41329-w.

[4] Z. Zheng *et al*, "Towards an understanding of large language models in software engineering tasks," *Empir. Softw. Eng.*, vol. 30, no. 2, p. 50, 2024, doi: 10.1007/s10664-024-10602-0.

[5] S. Saleem, M. N. Asim, L. Van Elst, and A. Dengel, "Generative language models potential for requirement engineering applications: insights into current strengths and limitations," *Complex & Intelligent Systems*, vol. 11, no. 6, p. 278, 2025, doi: 10.1007/s40747-024-01707-6.

[6] J. Zhang, J. Zhou, N. Niu, J. Hua, and C. Liu, "Synergistic enhancement of requirement-to-code traceability: A framework combining large language model based data augmentation and an advanced encoder," *Inf. Softw. Technol.*, vol. 193, p. 108045, 2026, doi: 10.1016/j.infsof.2026.108045.

[7] A. Majidzadeh, M. Ashtiani, and M. Zakeri-Nasrabadi, "Multi-type requirements traceability prediction by code data augmentation and fine-tuning MS-CodeBERT," *Comput. Stand. Interfaces*, vol. 90, p. 103850, 2024, doi: 10.1016/j.csi.2024.103850.

[8] A. Ferrari and P. Spoletini, "Formal requirements engineering and large language models: A two-way roadmap," *Inf. Softw. Technol.*, vol. 181, p. 107697, 2025, doi: 10.1016/j.infsof.2025.107697.

[9] F. R. Arnob, R. H. Mollik, P. Goyal, and R. Bryce, "Bug Triaging Based on Transformer Models Utilizing Commit Messages," in *The 22nd International Conference on Information Technology-New Generations (ITNG 2025)*, S. Latifi, Ed., Cham: Springer Nature Switzerland, 2025, pp. 354–366.

[10] K. Ronanki, B. Cabrero-Daniel, and C. Berger, "ChatGPT as a Tool for User Story Quality Evaluation: Trustworthy Out of the Box?," in *Agile Processes in Software Engineering and Extreme Programming – Workshops*, P. Kruchten and P. Gregory, Eds., Cham: Springer Nature Switzerland, 2024, pp. 173–181.

[11] J. Zhang, J. Zhou, N. Niu, J. Hua, and C. Liu, "Synergistic Enhancement of Requirement-to-Code Traceability: A Framework Combining Large Language Model based Data Augmentation and an Advanced Encoder," Oct. 2025, [Online]. Available: http://arxiv.org/abs/2509.20149

[12] B. Wang, Z. Zou, X. Liang, H. Jin, and P. Liang, "HGNNLink: recovering requirements-code traceability links with text and dependency-aware heterogeneous graph

neural networks," *Automated Software Engineering*, vol. 32, no. 2, p. 55, 2025, doi: 10.1007/s10515-025-00528-2.

[13] S. Rezig, Z. Achour, and N. Rezg, "Retraction:Using data mining methods for predicting sequential maintenance activities," Nov. 07, 2018, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/app8112184.

[14] A. Majidzadeh, M. Ashtiani, and M. Zakeri-Nasrabadi, "Multi-type requirements traceability prediction by code data augmentation and fine-tuning MS-CodeBERT," *Comput. Stand. Interfaces*, vol. 90, p. 103850, 2024, doi: 10.1016/j.csi.2024.103850.

[15] A. Vogelsang and J. Fischbach, "Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline," May 2024, [Online]. Available: http://arxiv.org/abs/2402.13823

[16] Z. Zheng *et al.*, "Towards an understanding of large language models in software engineering tasks," *Empir. Softw. Eng.*, vol. 30, no. 2, p. 50, 2024, doi: 10.1007/s10664-024-10602-0.

[17] N. Marques, R. R. Silva, and J. Bernardino, "Using ChatGPT in Software Requirements Engineering: A Comprehensive Review," *Future Internet*, vol. 16, no. 6, 2024, doi: 10.3390/fi16060180.

[18] N. N. Tabari, "Automated Agile User Story Quality Assessment with Natural Language Processing," 2024.

[19] A. Sharma and A. Kumar Tripathi, "Evaluating user story quality with LLMs: a comparative study," *J. Intell. Inf. Syst.*, vol. 63, no. 4, pp. 1423–1451, 2025, doi: 10.1007/s10844-025-00939-3.

[20] T. F. Boka, Z. Niu, and R. B. Neupane, "A survey of sequential recommendation systems: Techniques, evaluation, and future directions," *Inf. Syst.*, vol. 125, p. 102427, 2024, doi: 10.1016/j.is.2024.102427.

[21] C. Ellsel and R. Stark, "Advancing Requirements Engineering with Large Language Models," *Procedia CIRP*, vol. 136, pp. 701–706, 2025, doi: 10.1016/j.procir.2025.08.120.

[22] A. Hemmat, M. Sharbaf, S. Kolahdouz-Rahimi, K. Lano, and S. Y. Tehrani, "Research directions for using LLM in software requirement engineering: a systematic review," *Front. Comput. Sci.*, vol. 7, 2025, doi: 10.3389/fcomp.2025.1519437.

[23] X. Du, Z. Liu, C. Li, X. Ma, Y. Li, and X. Wang, "LLM-BRC: A large language model-based bug report classification framework," *Software Quality Journal*, vol. 32, no. 3, pp. 985–1005, 2024, doi: 10.1007/s11219-024-09675-3.