

# Transaction Categorization with Relational Deep Learning in QuickBooks

Kaiwen Dong<sup>1,2</sup>, Padmaja Jonnalagedda<sup>1</sup>, Xiang Gao<sup>1</sup>, Ayan Acharya<sup>1</sup>, Maria Kissa<sup>1</sup>, Mauricio Flores<sup>1</sup>, Nitesh V. Chawla<sup>2</sup>, and Kamalika Das<sup>1</sup> (✉)

<sup>1</sup> Intuit, Mountain View CA 94043, USA

{kaiwen\_dong,saisri\_jonnalagedda,Xiang\_Gao,maria\_kissa,  
mauricio\_flores,Kamalika\_Das}@intuit.com

<sup>2</sup> University of Notre Dame, Notre Dame IN 46556, USA

{kdong2,nchawla}@mus@nd.edu

**Abstract.** Automatic transaction categorization is crucial for enhancing the customer experience in QuickBooks by providing accurate accounting and bookkeeping. The distinct challenges in this domain stem from the unique formatting of transaction descriptions, the wide variety of transaction categories, and the vast scale of the data involved. Furthermore, organizing transaction data in a relational database creates difficulties in developing a unified model that covers the entire database. In this work, we develop a novel graph-based model, named **Rel-Cat**, which is built directly over the relational database. We introduce a new formulation of transaction categorization as a link prediction task within this graph structure. By integrating techniques from natural language processing and graph machine learning, our model not only outperforms the existing production model in QuickBooks but also scales effectively to a growing customer base with a simpler, more effective architecture without compromising on accuracy. This design also helps tackle a key challenge of the cold start problem by adapting to minimal data.

**Keywords:** Transaction Categorization · Relational Deep Learning · Financial Applications.

## 1 Introduction

QuickBooks offers essential bookkeeping and accounting capabilities tailored to the needs of small and medium-sized businesses. It enables them to efficiently manage critical aspects of their business operations, including accounting, payroll, payments, and inventory. A key feature of QuickBooks is its ability to categorize financial activities captured through invoice descriptions, bank statements, etc. flexibly to enhance insights into business performance and to streamline tax compliance. By automating labor-intensive and error-prone tasks, QuickBooks allows business owners to focus on driving growth and increasing revenue.

At the core of QuickBooks’s functionality is its advanced bookkeeping experience. Most business transactions today are processed through financial institutions, and QuickBooks integrates seamlessly with these institutions, enabling

businesses to link their accounts and synchronize data. This connectivity triggers an influx of transactions—approximately 6.2 billion annually into QuickBooks. Having business owners or accountants manually review transactions would be an ineffective use of their time. Automating the processing of such a vast volume of transactions is crucial.

To facilitate its sophisticated accounting features, QuickBooks organizes transactions into specific categories or accounts. For example, a fuel purchase at an ExxonMobil station might be categorized under “Transportation”, while an electricity bill could be classified as “Utilities”. This paper addresses QuickBooks’s transaction categorization challenge, employing state-of-the-art methods in natural language processing [4] and graph machine learning [3] to solve this as a link prediction problem [5] within a relational database. Drawing inspiration from Relational Deep Learning [8], we propose a unified approach to effectively model the transaction database through interconnected relational tables, introducing modifications specific to the unique challenges and practical requirements of QuickBooks.

### 1.1 Problem statement

In QuickBooks, effective bookkeeping relies on the accurate categorization of transactions into specific accounts. Businesses have unique needs and preferences for how transactions are categorized. QuickBooks facilitates this by allowing customization of account names. This feature enables different companies to maintain both common and distinct account names based on their individual requirements. To further refine the organization of financial data and support compliance with tax regulations, QuickBooks allows users to classify these account names into a structured hierarchy of more abstract account types. This system not only personalizes the accounting experience for each business but also guarantees the accurate recording of financial activities. The more abstract account type, referred to as *Code*, corresponds to the IRS tax code, while the more granular account name, *Category*, is user-defined. For example, *Category* “Airfare” and “Internet” can both be grouped into a more abstract *Code* “Business Expenses”.

The primary task we address in this paper is predicting the appropriate *Category* for any new transaction imported into QuickBooks. In addition to delivering the most likely categorization for a new transaction, we explore providing the Top-5 probable Categories. This approach is predicated on the likelihood that the company’s preferred *Category* is more often found within the Top-5 predictions rather than solely the top prediction. Offering a selection of probable *Category* can significantly enhance user trust in QuickBooks’s capabilities, fostering a stronger reliance on QuickBooks for critical financial management tasks.

The data supporting this transaction categorization task is managed within a relational database, where various tables are interconnected through primary and foreign keys. The database schema, detailed in Figure 1a, includes the following critical tables:

1. **Transaction table:** Stores records of all transactions across different companies.
2. **Category table:** Contains all specific account names used by QuickBooks users.
3. **Code table:** Includes the abstract account types aligned with overarching tax codes, that facilitate the organization of *Category*.
4. **Company table:** Contains information about companies utilizing QuickBooks.

## 1.2 Related Works

Transaction categorization is fundamental to the user experience in QuickBooks. There are several ways to solve and productionize transaction categorization in accounting systems. To date, two major approaches have been deployed to effectively solve the task in QuickBooks.

The first approach, known as IRIS [12], categorizes incoming transactions based solely on a company’s historical data. It begins by extracting business entity names from transaction descriptions using a rule-based normalization process that includes case folding and digit folding. IRIS then queries the company’s historical transactions for similar business entities, defining similarity via Jaccard similarity—entities are considered similar if they are frequently categorized together. A weighted voting mechanism is then employed to determine the most likely *Category* for the new transaction. While IRIS is an efficient system capable of managing large datasets and ensuring quick database queries, its reliance on a company’s historical data limits its utility, especially for new users with little to no transaction history.

The second methodology is embodied in the work by [13] in QuickBooks. This work addresses the limitations of IRIS by enhancing performance for users new to the system (cold-start users) with a populational model. It utilizes a Word2Vec-based [14] encoding for transactions and *Category*, and applies a contrastive learning framework to maximize the matching pairs of transaction-*Category* and minimize the non-matching pairs. It also employs a logistic regression classifier to enable personalized categorization for each company. During inference, the calibrated population and personalized model is applied to predict the category of a new transaction. Although this method has demonstrated effective performance in practice, maintaining an individual logistic regression classifier for each company introduces significant overhead and risks of overfitting. Additionally, calibrating two models can lead to suboptimal performance due to challenges in effectively leveraging strengths of both models.

## 1.3 Challenges and Contributions

*Challenges.* The transaction categorization task in QuickBooks presents several core challenges. Firstly, transaction data is stored in a relational database composed of multiple tables, making it difficult to effectively apply a single unified machine learning model without explicitly engineering features to consolidate

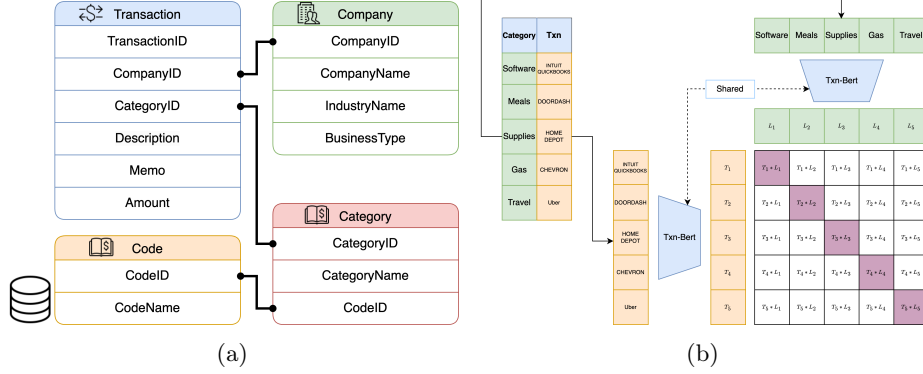


Fig. 1: (a) The schema of the relational database of QuickBooks transactions. (b) Training paradigm of Txn-Bert.

the tables. Secondly, the transaction description field, crucial for identifying the semantic meaning of a transaction, often follows the formatting standards of financial institutions rather than natural language. This necessitates an effective method to encode transaction data, capturing nuances such as business entity names and financial acronyms. Thirdly, the vast scale and skewed distribution of *Category* labels result in a highly imbalanced learning scenario. The categorization is highly personalized so that one transaction can be put into different *Category* by different users. Traditional multi-class classifiers can struggle with effectiveness and generalizability on this task. Last but not the least, the large volume of transactions processed by QuickBooks requires a balance between model capability and computational efficiency to ensure real-time performance.

*Contributions.* The design choices of the proposed pipeline are geared towards tackling one or more of these challenges. The salient contributions are summarized as follows:

1. To model the relational database of the transaction data, we transform the database to a heterogeneous graph and apply a unified graph model, **Rel-Cat**, to effectively represent the relationships within the data and support both new and old users of QuickBooks.
2. To encode the transaction data with unique linguistic characteristics, we integrate a trained-from-scratch text encoder **Txn-Bert** into **Rel-Cat**, which effectively captures the semantics of the transaction data.
3. To handle large-scale transaction data efficiently, we introduce practical techniques to improve **Rel-Cat**'s scalability, including a similarity-based neighbor sampling, edge direction dropping, and Top-K Nearest Neighbor early exit.

## 2 Txn-Bert: Text Encoder trained from scratch

In this section, we introduce **Txn-Bert**, a text encoder developed specifically for encoding transaction descriptions into fixed-length embeddings. The entire encoder is trained from scratch, recognizing the unique linguistic patterns found in transaction data. We begin with the rationale behind pretraining a new language model, followed by the detailed process of the training of the transformer model.

### 2.1 Unique Linguistic Characteristics of Transaction Data

To effectively convert transaction text data into embeddings, it is essential to first examine the unique characteristics of this data.

Transaction descriptions, found in banking statements, serve as crucial identifiers for financial activities within businesses. These descriptions adhere to formatting standards set by financial institutions. For instance, Visa mandates that business entity names in transaction descriptions be no longer than 25 characters, requiring abbreviations as necessary.<sup>3</sup>

Such descriptions, therefore, consist predominantly of abbreviated business names, a feature markedly distinct from the more fluid and expansive natural language. This distinctiveness necessitates a specialized approach for encoding transaction. The conventional language models, typically pretrained on generic natural language datasets, do not suffice for capturing the nuances of transaction descriptions. This inadequacy forms the core motivation for developing **Txn-Bert** from the ground up, ensuring it is finely tuned to the specific lexicon and syntax of transaction language.

### 2.2 Training the transformer model

We start by building a custom tokenizer to fit the specific vocabulary of transaction descriptions. Once we have built the custom tokenizer, the next step is to train our text encoder. We train the model to classify new transactions into specific *Category* solely based on the information from that transaction—like its description, amount, and any additional memo—without considering the historical patterns of the company. We model our training approach after Sentence-Bert [17], treating it as a sentence pair matching problem to match transactions to *Category*. The training paradigm of **Txn-Bert** is shown in Figure 1b. We adopt a Siamese network architecture, where a shared text encoder independently processes a pair of a transaction and its corresponding *Category*.

We format each transaction by combining its key fields into a single sentence. We include the <description>, <amount>, and <memo>. We also add a <polarity> field, which labels the transaction as “received” if the amount is positive or “paid” if it’s negative. The complete transaction text looks something like

<sup>3</sup> <https://usa.visa.com/content/dam/VCOM/download/merchants/visa-merchant-data-standards-manual.pdf>

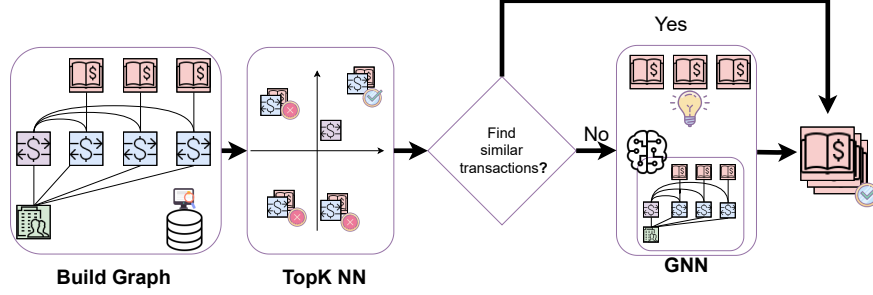


Fig. 2: The overview pipeline of Rel-Cat.

this: “Transaction <polarity> \$<amount> for: <description> <memo>.” For the *Category* labels, we use them just as they are.

The text encoder is a transformer [18], which will refine the representation of each token iteratively through the transformer blocks. After processing through these layers, we take a mean pooling strategy to get a single representation for the whole sentence.

The training objective mirrors the CLIP approach [16], optimizing for high cosine similarity between matched transaction-*Category* pairs and low similarity for unmatched pairs. We use a symmetric cross-entropy loss. Specifically, given a batch of  $N$  transaction-*Category* pairs  $\{(T_i, L_i) | 1 \leq i \leq N\}$ , and our text encoder  $f(\cdot)$ , the training loss is:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \log \left( \frac{e^{\text{Sim}(f(T_i), f(L_i))}}{\sum_{j=1}^N e^{\text{Sim}(f(T_i), f(L_j))}} \right) + \log \left( \frac{e^{\text{Sim}(f(T_i), f(L_i))}}{\sum_{j=1}^N e^{\text{Sim}(f(T_j), f(L_i))}} \right)$$

where  $\text{Sim}(v, w) = \frac{v \cdot w}{|v| \cdot |w|}$  represents the cosine similarity. This method leverages all non-matching pairs within a batch for contrastive learning, thus eliminating the need for explicit negative sampling. This effective training allows Txn-Bert to robustly encode transaction and *Category* data for use in downstream tasks.

### 3 Rel-Cat: Modeling relations within database

In this section, we introduce **Rel-Cat**. We outline our approach for preprocessing a relational database into a heterogeneous graph, thereby redefining the transaction categorization task as a link prediction problem within this graph. We then describe **Rel-Cat**, a hybrid method combining rule-based early exit (TopK NN) with a robust heterogeneous graph neural network (GNN). The overview pipeline can be found in Figure 2.

#### 3.1 Build a heterogeneous graph from a relational database

The transaction data in QuickBooks is organized within a relational database comprising multiple interconnected tables, each representing different facets of

the transaction data. To unify modeling across this multi-table architecture, we employ the concept of Relational Deep Learning [8], transforming the relational database into a heterogeneous graph. We provide an overview of this transformation process with a conversion diagram depicted in Figure 3.

**Conversion overview** Referencing Figure 1a, the relational database for transaction data consists of four key components:

1. **Multiple tables:** The database is structured into several tables, each detailing a specific aspect of the transactions.
2. **Rows within tables:** Each table comprises multiple rows, each row encapsulating a distinct transaction or fact.
3. **Foreign-primary key relationships:** Rows across different tables are interconnected via foreign-primary key associations, facilitating relational references among them.
4. **Row attributes:** Each row includes attributes that describe its elements, such as the transaction description in the transaction table or the company name in the company table.

These elements of the relational database are mapped to the components of a heterogeneous graph as follows:

1. **Node types:** Each table in the database is treated as a distinct node type within the graph.
2. **Nodes:** Individual rows within a table are represented as nodes with the node type corresponding to their table.
3. **Edges:** Connections between rows (nodes) across tables, facilitated by foreign-primary key pairs, are represented as edges in the graph.
4. **Node attributes:** Attributes of each row, particularly textual data, are utilized as node attributes in the graph.

Following the method described in [8], we convert a relational database  $(\mathcal{T}, \mathcal{L})$  into a heterogeneous graph  $G = (\mathcal{V}, \mathcal{E}, \phi, \psi)$ . The relational database consists of tables  $\mathcal{T} = \{T_i\}$ , each containing rows that become graph nodes  $v \in \mathcal{V}$ , with node types assigned by table origin via  $\phi$ . Attributes  $x_v$  of each row  $v$  are encoded using **Txn-Bert** embeddings. Edges  $\mathcal{E}$  between nodes represent primary-foreign key relationships across tables defined by links  $\mathcal{L}$ , with relation types  $\mathcal{R} = \mathcal{L} \cup \mathcal{L}^{-1}$  accounting for both original and inverse relations via  $\psi$ .

**Transform to a link prediction task** In the heterogeneous graph we’ve constructed, historical transaction nodes  $v \in T_{\text{transaction}}$ , that have already been categorized form links with corresponding *Category* nodes, such that  $p_v \in \mathcal{K}_v$  where  $v' \in T_{\text{Category}}$ . However, new transactions that have not yet been categorized enter the graph without any existing links to *Category* nodes ( $p_{v'} \notin \mathcal{K}_v$  where  $v' \in T_{\text{Category}}$ ). This scenario redefines the transaction categorization task. Instead of assigning a *Category* to each new transaction, our task shifts to

predicting which *Category* node in the graph should be linked to the new transaction node. Essentially, the categorization challenge is transformed into a link prediction task as shown in Figure 3a, where the objective is to determine the most appropriate connections for uncategorized transaction nodes based on the graph’s existing structure and the attributes of its nodes. This design also allows us to address the cold start problem, a key challenge for our task, by mapping a transaction from a new user to a relevant *Category* based on the historical similarity from other similar businesses.

### 3.2 Model the heterogeneous graph

This section details how **Rel-Cat** models the heterogeneous graph  $G$ . We employ a message-passing Graph Neural Network (GNN) [10], specifically a variant of GraphSAGE [11], to encode node representations. Given the graph’s diverse node and edge types, we adapt our approach to model message-passing along different edge types separately. Each message type is initially processed through a homogeneous GNN. Messages from various edge types are then combined using a second-level aggregation function to get a comprehensive node representation:

$$\mathbf{h}_v^{(i+1)} = t_{\phi(v)} \left( \mathbf{h}_v^{(i)}, \text{AGG}_{\text{Heter}}(\{ \text{AGG}_{\text{Homo}}(\{ g_R(\mathbf{h}_w^{(i)}) \mid w \in \mathcal{N}_R(v) \}) \mid \forall R = (T, \phi(v)) \in \mathcal{R} \}) \right), \quad (1)$$

where  $\mathcal{N}_R(v) = \{w \in \mathcal{V} \mid (w, v) \in \mathcal{E} \text{ and } \psi(w, v) = R\}$  is the neighborhood of node  $v$  under the specific edge type  $R$ . Here,  $\text{AGG}_{\text{Homo}}$  employs a mean operator to normalize message contributions within the same type, while  $\text{AGG}_{\text{Heter}}$  uses attention mechanism [18] to dynamically weight the importance of different message types, improving the model’s prioritization of relevant messages

**Two-hop connections for transaction nodes** Our initial node representation learning framework in **Rel-Cat**, as outlined in Equation 1, aggregates messages from immediate neighbors within the graph  $G$ . While effective, this approach can introduce issues inherent in GNNs such as over-squashing [1] and limited expressiveness [20], which we address through graph data augmentation.

*Graph data augmentation.* When GNNs learn the node representation of the **target transaction** in Figure 3b, they have to propagate at least two rounds of message passing to receive the information from the **historical transactions** of the same company. To enhance node representation, especially for the target transaction node, we introduce an additional type of edge within transaction nodes, effectively making historical transactions direct neighbors of the target transaction. This adjustment allows the target transaction to aggregate messages from historical transactions in just one message-passing iteration. The augmented graph with these new connections is shown in Figure 3c. For this new edge type (transaction to transaction), we utilize the GATv2 [2] as the aggregation operator ( $\text{AGG}_{\text{Homo}}$ ). While applying GATv2 to all edge types can lead to GPU



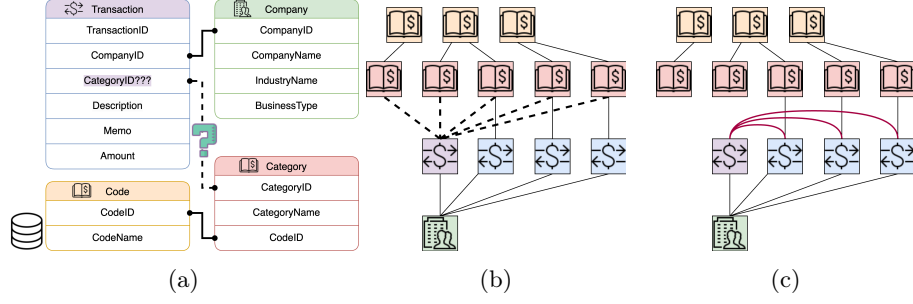


Fig. 3: Transformation of a relational database into a heterogeneous graph for transaction categorization. (a) A new transaction enters the system without a foreign key connection to the *Category* table. (b) The heterogeneous graph is built from the relational database, where transaction categorization is formulated as a link prediction task. (c) Two-hop connections for transaction nodes mitigate over-squashing and improve model expressiveness.

memory issues, using it for this specific edge type does not significantly increase memory requirements.

Specifically, we introduce an extra set of edges  $\mathcal{E}_{\text{aug}}$  into the original graph:

$$\mathcal{E}_{\text{aug}} = \{(v_1, v_2) \in \mathcal{V} \times \mathcal{V} \mid v_1, v_2 \in T_{\text{transaction}}, \exists v_c \in T_{\text{company}}, p_c \in \mathcal{K}_{v_1} \cap \mathcal{K}_{v_2}\}.$$

The edge set  $\mathcal{E}_{\text{aug}}$  is the set of transaction pairs if they are from the same company. Next, we discuss how this modification can mitigate the two issues.

*Addressing over-squashing.* GNNs learn the node representation iteratively from the local neighborhood. However, due to this recurrent learning paradigm, GNNs often face a challenge where the information from a growing receptive field is compressed into a fixed-length vector, potentially losing valuable information. This phenomenon is prevalent in the transaction graph  $G$ . For the **target transaction** in Figure 3b, the message from its **historical transactions** is equally compressed into one fixed-length vector. Since this message is not conditioned on the target transaction, it cannot adjust itself so that the signals more important to the target transaction are preserved. By rewiring the graph and applying a weighted message aggregator like GATv2, it can effectively enable the target transaction’s incoming message to keep the most relevant information [1], enhancing the effectiveness of the prediction.

*Enhancing expressiveness.* GNNs essentially simulate the Weisfeiler-Lehman graph isomorphism algorithm [20]. This limits their expressiveness in terms of distinguishing non-isomorphic graphs. By directly connecting two-hop neighbors (historical transactions) as immediate neighbors to the target transaction in graph  $G$ , our model approximates a K-hop GNN approach [7]. This augmentation not only improves expressiveness but does so without significantly increasing computational costs, thus maintains a balance between accuracy and efficiency.

### 3.3 Training objective

The transformation of the relational database into the heterogeneous graph  $G$  redefines our task as a link prediction challenge. In essence, link prediction in this context is a ranking problem where the model is expected to rank the correct node pair (the ground truth link) higher than other node pairs (non-connected links). Specifically, for a transaction node  $v_i$  and its corresponding *Category* node  $v_j$  in graph  $G$ , the score of  $(v_i, v_j)$  should be higher than that of any other *Category* node pair  $(v_i, v_k)$ , where  $v_k$  represents any *Category* node not connected to  $v_i$ . We utilize the inner product as the scoring function between transaction and *Category* nodes, employing AUCLoss [19], a surrogate for AUC, as the training objective of **Rel-Cat**:

$$\mathcal{L} = \sum_{(v_i, v_j) \in \mathcal{E}^+, (v_i, v_k) \in \mathcal{E}^-} (1 - (\mathbf{h}_i * \mathbf{h}_j) + (\mathbf{h}_i * \mathbf{h}_k))^2, \quad (2)$$

where  $\mathbf{h}_v$  denotes the final representation of either a transaction or *Category* node from Equation 1. The set of positive node pairs  $\mathcal{E}^+$  is chosen as:

$$\mathcal{E}^+ \subseteq \{(v_i, v_j) \in T_{\text{transaction}} \times T_{\text{Category}} \mid p_{v_j} \in \mathcal{K}_{v_i}\}. \quad (3)$$

The set of negative node pairs,  $\mathcal{E}^-$ , includes non-connected links, defined as:

$$\mathcal{E}^- \subseteq \{(v_i, v_k) \in T_{\text{transaction}} \times T_{\text{Category}} \mid p_{v_j} \notin \mathcal{K}_{v_i}\}. \quad (4)$$

**Weighted negative sampling** The set of negative node pairs,  $\mathcal{E}^-$ , consists of non-connected links. Due to the impracticality of enumerating all potential negative pairs, we employ a negative sampling strategy. For a transaction node  $v_i$  and its corresponding positive *Category* node  $v_j$ , simple uniform sampling from  $T_{\text{Category}} \setminus \{v_j\}$  would be suboptimal due to the long-tail distribution of *Category*. This could prevent effective learning if those *Category* nodes  $v_k$ , which rarely appear as positives, are sampled as negatives. To address this, we employ a weighted multinomial distribution for negative sampling, where weights are assigned proportional to the normalized frequency of *Category* appearances, enhancing the relevance and challenge of the sampled negatives.

**Selective Loss Computation via Diversity Filtering** During the loss computation and backpropagation over a batch - not all samples contribute equally to learning, some being too easy (providing little gradient signal) or redundant (leading to inefficient updates). Thus, we propose a selective loss computation strategy, where all samples undergo a forward pass, but only a subset of challenging or informative samples contribute to the loss and gradient computation, i.e., a full-forward partial-backward strategy. This strategy retains the benefits of large-batch inference while ensuring that gradient updates focus on samples that are most challenging and diverse. Since all samples contribute to feature extraction during the forward pass, the model benefits from seeing a broader

distribution of the data. This results more accurate hard sample identification, targeted gradient updates, and improved generalization. To select the subset of diverse samples, we compute the dot product similarity of transaction feature representation. From this distribution, we gradually refine the selection over epochs, choosing samples with lower similarity scores and only use those for loss computation. Initially, we utilize 100% of the samples, then gradually decrease this proportion in stages until reaching 40% , maintaining a core set of diverse samples throughout training. Experiments show that this provides us with the optimal diversity samples for loss computation in a batch, providing highest gradient signals during back-propagation.

### 3.4 Scalability and practical designs

In this section, we detail several scalability improvements to **Rel-Cat**, designed to effectively manage the extensive volume of transaction data encountered in real-world applications. We discuss practical designs to reduce neighborhood sizes for transaction and *Category* nodes during node representation encoding and introduce a rule-based early exit method, Top-K Nearest Neighbor, to efficiently manage the workload on GNNs.

**Reducing neighborhood size** Given that GNNs encode node representations by aggregating features from local neighborhoods, large neighborhoods can pose significant scalability challenges, particularly in terms of memory consumption. To address this, we implement a node-wise neighbor sampling strategy. Different from existing methods [11,9,21], we further take node types into consideration to optimize the receptive field of the GNNs. We employ distinct strategies for transaction nodes and *Category* nodes:

*Similarity-based neighbor sampling for transaction nodes.* For transaction nodes,  $\{v \mid v \in T_{\text{transaction}}\}$ , the local neighborhood typically includes all historical transactions associated with the company, which can be extensive. For example, there are companies owning over 2000 transactions within just one year. To manage this, we utilize similarity-based neighbor sampling, which reduces neighborhood size while preserving relevant information.

We compute the cosine similarity between the text embeddings of the target transaction node and its historical counterparts. Historical transactions are then sampled based on their similarity scores, prioritizing those most relevant to the target. This approach not only constrains computational overhead but also ensures that the most informative connections are maintained in **Rel-Cat**’s neighborhood. Techniques like Faiss [6] can be employed to enhance the efficiency of these computations.

*Edge direction dropping for Category nodes.* *Category* nodes,  $\{v \mid v \in T_{\text{Category}}\}$ , often have neighborhoods that include a huge set of transaction nodes linked to them. The popularity of some *Category* can lead to extremely large neighborhood sizes, which are not only impractical to process, but also ineffective to model.

To mitigate this, we discard all incoming edge connections from transaction to *Category* nodes. This adjustment significantly reduces the computational load by limiting the receptive field to exclude transaction nodes, while outgoing connections from *Category* to transaction nodes are kept. This approach not only simplifies the computation required to encode *Category* node but also ensures consistency in *Category* node representation during inference, as the computational graph remains unchanged regardless of new transactions being added.

**Top-K Nearest Neighbor** Despite the sophisticated capabilities of **Rel-Cat**, equipped with GNNs to perform predictions based on graph structure  $G$ , real-world scenarios often present varying degrees of prediction difficulty [15]. Many transactions imported into QuickBooks by users are very similar or even identical to past entries due to routine business activities. In such cases, users frequently reuse the same *Category* as in previous transactions. Consequently, a significant portion of transactions could be accurately categorized without necessitating full GNN processing. To capitalize on this, we utilize the streamlined and effective early exit method through Top-K Nearest Neighbor (TopK NN).

Upon the arrival of a new transaction, while we still prepare the graph  $G$  for subsequent GNN processing, we first assess if sufficiently similar historical transactions might already provide reliable predictions. Utilizing the similarity scores computed during the similarity-based neighbor sampling of transaction nodes, we identify a subset of historical transactions that exhibit a similarity score exceeding a cutoff, set at 0.8 for our experiments.

From this subset, we directly derive *Category* from the top- $K$  most similar transactions. Given that our categorization task aims to predict the 5 most likely categories, **Rel-Cat** will output these labels directly if 5 distinct *Category* are available from this subset. If fewer than 5 categories are available, the graph  $G$  is then processed by the GNNs to generate the remaining predictions.

## 4 Experiments

### 4.1 Experimental setup

**Dataset** To evaluate the performance of **Rel-Cat** comprehensively, we curated a dataset from active QuickBooks users as of November 2023. We randomly selected 7.5K companies and used their two most recent transactions with labeled *Category* post-November 2023 as our test set, resulting in a total of 15K transactions. The training set consisted of 3000K transactions having 100K *Category* across 15K companies reflecting a wide range of user preferences and patterns.

**Experimental settings** We benchmark the performance of **Rel-Cat** against the current production models in QuickBooks, namely **Shorthair** and **Lynx**. Shorthair is a population model that employs contrastive learning and Word2Vec embeddings, whereas Lynx, built on top of Shorthair is a logistic regression model

customized to a company. We assess the models using the metrics of Top-1, Top-2, and Top-5 accuracy, which measure whether the correct label is among the Top-k predictions of the model, ranked by the *Category* scores. We conduct evaluations under two distinct settings:

*Zero Shot:* In this setting, categorization is based solely on the information from the new transaction itself, without any contextual data from the owning company or its historical transactions. Both **Shorthair** and **Txn-Bert** are evaluated under this setting.

*Few Shot:* This setting incorporates not only the data from the new transaction but also contextual information from the owning company and its historical data. **Lynx**, **TopK NN**, and **Rel-Cat** are assessed under this framework.

## 4.2 Results

The experimental results are presented in Figure 1. In the Zero Shot setting, **Txn-Bert** outperforms the production **Shorthair** model significantly. The **Txn-Bert** model with 6 layers achieves a Top-1 accuracy boost of 7.76% and a Top-5 accuracy of 74.12%, indicating that our trained-from-scratch text encoder effectively captures the semantics of transaction descriptions and maps them accurately to the corresponding *Category*. The 12-layer **Txn-Bert** model shows only marginal improvement over the 6-layer model, suggesting that a lightweight language model pretrained from scratch is sufficient for encoding transaction data.

Table 1: Transaction categorization evaluated by accuracy under Zero Shot and Few Shot settings.

Methods	Top-1	Top-2	Top-5
<i>Zero Shot</i>			
Shorthair	36.07	-	-
Txn-Bert (6 layers)	43.83	57.96	74.12
Txn-Bert (12 layers)	<b>45.52</b>	<b>59.47</b>	<b>75.46</b>
<i>Few Shot</i>			
Lynx	62.49	-	-
TopK NN	65.80	73.63	78.55
Rel-Cat (GNNs only)	63.38	74.60	84.89
Rel-Cat	<b>68.67</b>	<b>78.97</b>	<b>88.04</b>
<i>Ablation Study</i>			
Rel-Cat (GNNs only)	63.38	74.60	84.89
w/o Txn-Bert	55.46	64.02	73.16
w/o two-hop connections	47.07	61.16	76.58
w/o similarity sampling	56.39	67.89	80.63
w/o diversity filtering	61.74	73.18	83.85

Table 2: Performance breakdown in different scenarios. **Acc** is the overall Top-1 accuracy, **HS** is the accuracy on Historical Seen subset, and **HU** is the accuracy on Historical Unseen subset.

Methods	Acc	HS	HU
TopK NN	65.80	<b>79.72</b>	0.16
Rel-Cat (GNNs only)	63.38	72.25	<b>22.05</b>
Rel-Cat	<b>68.67</b>	<u>78.64</u>	<u>20.84</u>
Methods (GNNs only)	Acc	HS	HU
Rel-Cat (GNNs only)	63.38	72.25	22.05
w/o Txn-Bert	55.46	66.24	4.83
w/o two-hop connections	47.07	51.36	26.93
w/o similarity sampling	56.39	65.26	14.73
w/o diversity filtering	61.74	70.46	20.75

In the Few Shot setting, **Rel-Cat** demonstrates substantial performance advantages. It achieves a Top-1 accuracy of 68.67% and a Top-5 accuracy of 88.04%,

significantly outperforming the **Lynx** model. Additionally, the **TopK NN** method, which uses text embeddings from **Txn-Bert**, achieves a Top-1 accuracy of 65.80%, surpassing the performance of **Rel-Cat** with GNNs alone. This result highlights the effectiveness of **TopK NN** in identifying recurring transactions in a user’s history, thereby enhancing Top-1 accuracy of **Rel-Cat**.

However, when the transaction categorization must be inferred beyond the user’s most similar historical transactions, the GNN component of **Rel-Cat** exhibits superior generalizability, achieving nearly 85% in Top-5 accuracy. This demonstrates that while **TopK NN** is highly effective for repeated transactions, the GNN module of **Rel-Cat** provides a broader and more accurate categorization capability for diverse transaction scenarios.

### 4.3 Seen vs Unseen Category in a Company’s History

In this section, we discuss why **Rel-Cat** is designed as a hybrid model, namely a combination of **TopK NN** and GNNs. In Table 2, we not only report our overall accuracy, but further break down the performance of **Rel-Cat** into Historical Seen and Historical Unseen scenarios. For Historical Seen, we choose the test samples such that their ground-truth *Category* is present within the company’s own history as context. For Historical Unseen, we choose the samples whose *Category* are present in the overall dataset but unseen to that company’s history.

We note that **TopK NN**, while having the best performance at repeated labels, has no predictive power for unseen labels. For this subset, **Rel-Cat**’s GNN is able to detect the labels with over 22% accuracy, highlighting its generalizable ability. Given that most transactions in production systems belong to the historical seen category, **Rel-Cat** has the best overall performance, balancing between both the subsets, thereby demonstrating the need for GNN and **TopK NN** hybrid model. Furthermore, analyzing these trends over the ablation studies, we note that each design choice adds to the overall accuracy, augmenting performance for either or both the seen and unseen subsets.

### 4.4 Ablation Studies

In this section, we delve deeper into the validation of the effectiveness of various design choices in **Rel-Cat**.

In Figure 1, we report ablation studies to validate if the proposed components in **Rel-Cat** can enhance the predictive power of transaction categorization. For the ablation studies, we only include the GNNs module for **Rel-Cat**.

*w/o Txn-Bert (Sec 2):* Replacing **Txn-Bert** with off-the-shelf Sentence-Bert [17], we observe a decrease in performance, underscoring the necessity of a trained-from-scratch text encoder tailored to transaction data. We further note the steep drop in performance for the unseen subset.

*w/o two-hop connections (Sec 3.2):* This ablation study emphasizes the critical role of explicit graph augmentation. It demonstrates that directly connecting the target and historical transactions as neighbors significantly enhances performance. This finding underscores that merely converting a relational database to

a heterogeneous graph without strategic modifications is inadequate for maximizing the model’s effectiveness. While this setting has the best performance for the unseen subset, it comes at the expense of significant decline in overall performance.

*w/o similarity sampling (Sec 3.4):* Testing the impact of neighbor sampling based on semantic similarity, we find that sampling similar historical transactions in GNN’s computation graphs allows **Rel-Cat** to leverage relevant information effectively for accurate predictions. We note that this setting also suffers from lack of generalizability.

*w/o diversity filtering (Sec 3.3):* here we report the results without the diversity filtering, when all samples pass through the backward pass during GNN training throughout the training. We note that while the drop in the unseen subset performance is to be expected without this filtering, we also note a drop across all metrics, suggesting the effectiveness of selective backpropagation.

#### 4.5 Time complexity

We have assessed the processing time for various components within **Rel-Cat**, with the results detailed in Table 3. We report the processing time as time taken in milliseconds (ms) per 1,000 transactions. The entire pipeline of **Rel-Cat** is designed to operate effectively on both CPU and GPU environments, catering to different production system requirements.

Table 3: Inference times for 1,000 transactions.

Walltime (ms)	CPU GPU	
<b>Txn-Bert</b>	1400	67
<b>Rel-Cat</b>	6808	843
- TopK NN	333	-
- <b>Rel-Cat</b> (GNNs only)	6614	324
<i>Total</i>	8208	910

On a CPU, the lightweight text encoder **Txn-Bert** processes 1,000 transactions in just 1400 milliseconds. To deliver the Top-5 predictions, **Rel-Cat** requires approximately 8 seconds for 1,000 transactions. In contrast, utilizing a GPU significantly reduces the processing time to under 1 second for outputting the Top-5 predictions. This efficiency demonstrates **Rel-Cat**’s capability to scale and meet the demands of real-world transaction volumes effectively.

#### 4.6 Prediction cascade

In the **Rel-Cat** pipeline shown in Figure 2, the processed graph initially enters the TopK NN module, which serves as an early exit strategy to identify similar

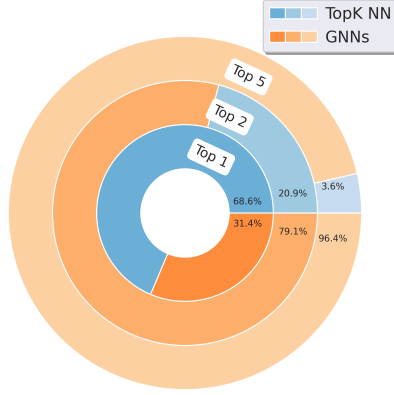


Fig. 4: Cascade process in **Rel-Cat**. TopK NN efficiently resolves 68% of transactions when only a Top-1 prediction is needed. However, for more comprehensive Top-5 predictions, over 96% of transactions necessitate processing by GNNs.

transactions via text embeddings. If this step does not yield sufficient *Category* predictions, the pipeline advances to the GNNs for additional predictions.

Our analysis of this cascade, shown in Figure 4, reveals that for Top-1 predictions, TopK NN alone efficiently handles over 68% of transactions. This high percentage underscores the prevalence of similar transactions within the database. However, to generate the Top-5 predictions, more than 96% of transactions require processing by the GNNs, indicating the need for a more in-depth computation to fulfill broader prediction requirements.

This flexible approach in **Rel-Cat** demonstrates the system’s adaptability, allowing for a balance between efficiency and thoroughness in prediction based on system demands and user experience considerations.

## 5 Conclusion

In this study, we introduce **Rel-Cat**, a unified model designed for transaction categorization within QuickBooks. Recognizing the unique linguistic characteristics of transaction data, we train, from scratch, a **Txn-Bert** text encoder, to grasp the semantic nuances of financial transaction descriptions. Subsequently, we employ a GNN-based model to capture the relationships among the tables in a relational database, redefining transaction categorization as a link prediction task over a heterogeneous graph. To address the challenges posed by the high volume of transactions and the specific demands of the categorization task, we integrate several innovative components that significantly boost **Rel-Cat**’s predictive capabilities. Our experimental results demonstrate that **Rel-Cat** not only surpasses previous production models in terms of performance but also



offers remarkable scalability and efficiency, making it well-suited for handling large-scale transaction data in real-world settings. Having proved its merit in offline testing, our approach is currently in the process of being implemented for deployment in production due to its improved metrics, improved customer experience providing multiple options for categories to choose from (top- $k$ ), and the simplified overall architecture allowing the company to move away from running and maintaining million plus models in production for supporting personalized transaction categorization.

**Acknowledgments.** We thank the anonymous reviewers for their insightful comments and helpful discussions. We are also grateful to Byron Tang, Heather Simpson, Jocelyn Lu, and Hilaf Hasson for their assistance in retrieving the dataset and providing constructive feedback on this project.

## References

1. Alon, U., Yahav, E.: On the Bottleneck of Graph Neural Networks and its Practical Implications. arXiv:2006.05205 [cs, stat] (Mar 2021), arXiv: 2006.05205
2. Brody, S., Alon, U., Yahav, E.: How Attentive are Graph Attention Networks? (Jan 2022), arXiv:2105.14491 [cs]
3. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine* **34**(4), 18–42 (Jul 2017), arXiv: 1611.08097
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019)
5. Dong, K., Guo, Z., Chawla, N.V.: Pure Message Passing Can Estimate Common Neighbor for Link Prediction (Oct 2023), arXiv:2309.00976 [cs]
6. Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazare, P.E., Lomeli, M., Hosseini, L., Jegou, H.: The Faiss library (2024), `_eprint: 2401.08281`
7. Feng, J., Chen, Y., Li, F., Sarkar, A., Zhang, M.: How Powerful are K-hop Message Passing Graph Neural Networks (Jan 2023), arXiv:2205.13328 [cs]
8. Fey, M., Hu, W., Huang, K., Lenssen, J.E., Ranjan, R., Robinson, J., Ying, R., You, J., Leskovec, J.: Position: Relational Deep Learning - Graph Representation Learning on Relational Databases (Jun 2024)
9. Frasca, F., Rossi, E., Eynard, D., Chamberlain, B., Bronstein, M., Monti, F.: SIGN: Scalable Inception Graph Neural Networks (Nov 2020), arXiv:2004.11198 [cs, stat]
10. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural Message Passing for Quantum Chemistry. *CoRR* (2017)
11. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive Representation Learning on Large Graphs. arXiv:1706.02216 [cs, stat] (Sep 2018), arXiv: 1706.02216
12. Lesner, C., Ran, A., Rukonic, M., Wang, W.: Large Scale Personalized Categorization of Financial Transactions. *Proceedings of the AAAI Conference on Artificial Intelligence* **33**(01), 9365–9372 (Jul 2019), number: 01

13. Liu, J., Pei, L., Sun, Y., Simpson, H., Lu, J., Ho, N.: Categorization of Financial Transactions in QuickBooks. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 3299–3307. KDD '21, Association for Computing Machinery, New York, NY, USA (Aug 2021)
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems*. vol. 26. Curran Associates, Inc. (2013)
15. Panda, P., Sengupta, A., Roy, K.: Conditional deep learning for energy-efficient and enhanced pattern recognition. In: Proceedings of the 2016 Conference on Design, Automation & Test in Europe. pp. 475–480. DATE '16, EDA Consortium, San Jose, CA, USA (Mar 2016)
16. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning Transferable Visual Models From Natural Language Supervision. In: Proceedings of the 38th International Conference on Machine Learning. pp. 8748–8763. PMLR (Jul 2021), ISSN: 2640-3498
17. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (Nov 2019)
18. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is All you Need. In: *Advances in Neural Information Processing Systems*. vol. 30. Curran Associates, Inc. (2017)
19. Wang, Z., Zhou, Y., Hong, L., Zou, Y., Su, H., Chen, S.: Pairwise Learning for Neural Link Prediction. arXiv:2112.02936 [cs] (Jan 2022), arXiv: 2112.02936
20. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How Powerful are Graph Neural Networks? CoRR **abs/1810.00826** (2018), arXiv: 1810.00826
21. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 974–983. KDD '18, Association for Computing Machinery, New York, NY, USA (Jul 2018)