



PhD-FSTM-2024-111

The Faculty of Sciences, Technology
and Medicine



Université Claude Bernard Lyon 1

École Doctorale **ED 512**

Informatique et Mathématiques
(InfoMaths)

DISSERTATION

Defence held on 16/12/2024 in Luxembourg

to obtain the degree of

**DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG
EN INFORMATIQUE**

AND

**DOCTEUR DE L'UNIVERSITÉ DE LYON
EN INFORMATIQUE**

by

FAROUK DAMOUN

Born on 28 November 1994 in Ksar El Kebir, Morocco

**Enhancing Federated Learning
for Financial Sector
via Graph Learning and Language Models**

Dissertation defence committee

Dr. Tegawendé F. BISSYANDÉ, Chairman
Associate Professor, University of Luxembourg

Dr. Radu STATE, Supervisor
Professor, University of Luxembourg

Dr. Omar CHERKAoui
Professor, University of Quebec in Montreal

Dr. Rémi BADONNEL, Vice-chair
Professor, University of Lorraine

Dr. Hamida SEBA, co-supervisor
Professor, University of Claude Bernard Lyon 1

Dr. Noura FACI
Professor, University of Claude Bernard Lyon 1

© FAROUK DAMOUN
ALL RIGHTS RESERVED, 2024



The work conducted within the context of this dissertation was supported by the Banque et Caisse d'Épargne de l'État (Spuerkeess) and the Luxembourg National Research Fund's (FNR) Industrial Fellowship programme under grant number 15829274, supplemented by ANR-20-CE39-0008.

DEDICATED TO
MY MOTHER, THE PROFESSOR

FATIMA OUCHRIF

الأستاذة فاطمة أوشريف

AND

MY FATHER, THE POET

NOUREDDINE DAMOUN

الشاعر نور الدين الدامون

Acknowledgments

Completing my PhD has been a long and challenging journey, one that would not have been possible without the support of many remarkable individuals.

First, I'm deeply grateful to Professor Radu State for his exceptional guidance and supervision, and his insightful research and industry initiatives have continuously inspired me throughout these four years, including all my colleagues from the SEDAN research group

I would also like to extend my sincere thanks to Professor Hamida Seba for co-supervising my work and for giving me the opportunity to pursue a cotutelle at Claude Bernard University Lyon 1 and hosting me at GOAL research group of the LIRIS research laboratory.

My heartfelt appreciation goes to the members of my thesis and defense committees. Thank you for dedicating your time and expertise to reviewing my work and for valuable feedback.

I'm also grateful to Valérie Gregoire for handling the university paperwork, allowing me to focus on my research, and to Jessica Giro for her constant support in setting up the cotutelle.

Last, but by no means least, I would like to thank my family. To my parents, for their endless love and encouragement, and to my brother, for his constant advice and support and to my lovely wife, Nadia, for listening patiently through every high and low, for your unconditional love, and for standing by me throughout this journey.

Farouk DAMOUN

Luxembourg, December 2024

Abstract

In the modern financial sector, the need for robust machine learning models is increasingly critical, yet privacy regulations and competitive concerns often make centralized data inaccessible. To overcome these challenges, this dissertation proposes several novel Federated Learning (FL) methodologies that enable institutions to collaboratively train models while addressing the critical trade-offs between privacy and data utility by integrating privacy-preserving mechanisms designed to prevent input recovery with minimal loss to data utility.

A key contribution of this research is the development of a federated learning framework for Privacy-Preserving Behavioral Anomaly Detection and fraud detection in financial transactions. By utilizing Graph Neural Networks (GNNs) on dynamic ego-centric graphs, the framework captures evolving transactional patterns to detect anomalies effectively, while preserving privacy. A novel domain-specific negative sampling technique enables model training without the need for labeled data from the federation participants, making it highly applicable in real-world scenarios. The results demonstrate that deep learning-based methods, particularly graph-level embedding, outperform traditional approaches in anomaly detection and improving fraud detection tasks, by introducing anonymization and noise-based mechanisms, even when the shared model gradients are exposed.

Additionally, we propose G-HIN₂Vec, a graph-level embedding technique for heterogeneous information networks, which models individuals, such as cardholders, using static and dynamic ego-centric graphs. This method serves as an anonymization mechanism that eliminates the need for personally identifiable information (PII) in federated models. By integrating Personalized Local Differential Privacy (PLDP), we provide an additional layer of

protection, ensuring that even in the event of a model breach, sensitive data remains secure.

Finally, the dissertation introduces the Federated Byte-Level Byte Pair Encoding (BPE) Tokenizer, a novel privacy-preserving tokenization approach designed for distributed textual datasets. This tokenizer outperforms existing models in vocabulary coverage and efficiency, while maintaining rigorous data privacy. Our federated tokenizer not only competes with centralized models but also demonstrates improvements in both text compression and privacy preservation, for both general and domain-specific tokenizers.

The methodologies presented in this dissertation, validated through real-world transaction and textual financial datasets, highlight the transformative potential of federated learning to enhance fraud detection and language model performance while preserving privacy of individuals and institutions through anonymization and noise-based privacy mechanisms.

Résumé

Dans le secteur financier moderne, la nécessité de modèles d'apprentissage automatique robustes devient de plus en plus cruciale, mais les réglementations sur la confidentialité et les préoccupations concurrentielles rendent souvent la centralisation des données impossible. Pour surmonter ces défis, cette thèse propose de nouvelles méthodologies de l'apprentissage fédéré (FL) permettant aux institutions de collaborer pour entraîner des modèles machine learning tout en abordant les compromis critiques entre la confidentialité et l'utilité des données, en intégrant des mécanismes de préservation de la confidentialité conçus pour empêcher la récupération des entrées avec une perte minimale d'utilité des données.

Une contribution clé de cette thèse est le développement d'un framework d'apprentissage fédéré pour la détection d'anomalies comportementales et la détection de la fraude dans les transactions financières. En utilisant des réseaux neuronaux de graphes (GNNs) sur des graphes dynamiques ego-centriques, qui permettent de capturer et de détecter les schémas transactionnels évolutifs afin de repérer les anomalies, tout en préservant la confidentialité des individus. Une nouvelle technique d'échantillonnage négatif spécifique au domaine permet l'entraînement du modèle sans la nécessiter de données étiquetées de la part des participants à la fédération, ce qui le rend applicable dans des scénarios industriels. Les résultats montrent que les méthodes basées sur l'apprentissage profond, en particulier les GNNs, surpassent les approches traditionnelles dans la détection d'anomalies et améliorent la détection des fraudes dans les données transactionnelles, en introduisant des mécanismes d'anonymisation et de bruit, même lorsque les gradients des modèles fédérés sont exposés.

De plus, nous proposons G-HIN₂Vec, une technique basée sur les réseaux neuronaux de

graphes pour les réseaux d'information hétérogènes, qui modélise des individus, tels que les détenteurs de cartes bancaires, en utilisant des graphes ego-centriques statiques et dynamiques. Cette méthode sert de mécanisme d'anonymisation qui élimine la nécessité d'utiliser un identifiant individuel, tel que les informations personnellement identifiables (PII), dans les modèles fédérés. En intégrant la confidentialité différentielle locale personnalisée (PLDP), nous fournissons une couche de protection supplémentaire, garantissant que même en cas de violation du modèle, les données sensibles restent sécurisées.

Enfin, la thèse introduit le tokenizer fédéré basé sur le codage par paires de caractères (BPE) au niveau du byte (Byte-level), une approche de tokenisation respectant la confidentialité des individus mentionnés dans les données textuelles sous forme de PII, conçue pour les ensembles de données textuelles distribuées. Ce tokenizer surpasse les modèles existants en termes de couverture du vocabulaire et d'efficacité, tout en maintenant une stricte confidentialité des données. Notre tokenizer fédéré est non seulement concurrentiel par rapport aux modèles centralisés, mais démontre également des améliorations en matière de compression de texte et de préservation de la confidentialité, pour les tokenizers généraux et spécifiques au domaine.

Les méthodologies présentées dans cette thèse, validées à l'aide de bases de données financières réelles, publiques et privées, transactionnelles et textuelles, mettent en évidence le potentiel de l'apprentissage fédéré pour améliorer la détection de la fraude et les performances des modèles de langage tout en préservant la confidentialité des individus et des institutions grâce à des mécanismes d'anonymisation et de confidentialité basés sur le bruit.

Contents

DEDICATION	4
ACKNOWLEDGMENTS	5
ABSTRACT	6
RÉSUMÉ	7
I INTRODUCTION	15
1.1 Context	16
1.2 Contributions and Organization	19
1.3 Dissemination	23
I Background	25
2 CONCEPTS & RELATED WORK	26
2.1 Federated Learning Fundamentals	27
2.2 Graph Machine Learning	32
2.3 Language Modeling	41
II Contributions	50
3 UNSUPERVISED REPRESENTATION LEARNING FOR HETEROGENEOUS GRAPHS IN CREDIT CARD TRANSACTIONS	51
3.1 Introduction	53
3.2 Related Studies	55
3.3 Problem statement	57
3.4 Our approach	62

3.5	Design of experiments	70
3.6	Experiments and results	74
3.7	Conclusion	78
4	FEDERATED LEARNING FRAMEWORK FOR PRIVACY-PRESERVING ANOMALY DETECTION IN FINANCIAL TRANSACTIONS	79
4.1	Introduction	81
4.2	Related Studies	83
4.3	Problem statement	84
4.4	Our approach	87
4.5	Design of experiments	96
4.6	Experiments and results	99
4.7	Conclusion	107
5	FEDERATED BYTE-LEVEL BPE TOKENIZER: A PRIVACY-PRESERVING APPROACH FOR DISTRIBUTED LANGUAGE TOKENIZATION	109
5.1	Introduction	111
5.2	Related Studies	112
5.3	Problem statement	114
5.4	Our approach	116
5.5	Design of experiments	124
5.6	Experiments and results	127
5.7	Conclusion	135
6	CONCLUSION AND FUTURE WORK	137
6.1	Conclusion	138
6.2	Future work	139
	APPENDIX A LANGUAGE MODEL TOKENIZER PERFORMANCE IMPROVEMENT ACROSS MODELS	142
	BIBLIOGRAPHY	147

List of Figures

2.1	Illustration of graph convolution operations. (*)	37
3.1	Transformation of credit card transactions into an ego-centric graph, layering graphs and metapaths to capture financial interaction patterns.	60
3.2	Ego-centric cardholder heterogeneous graph schema	60
3.3	G-HIN ₂ VEC training data generation process	69
3.4	Probability distribution of log graph sizes and assortativity coefficient.	76
3.5	t-SNE visualization of graph embeddings, with 16 clusters.	77
4.1	Federated anomaly detection with GNNs, sampling subgraphs, extracting features, and scoring anomalies.	88
4.2	Categorization of credit card fraud types based on graph configuration manipulations, illustrating how different alterations in topology and node features correspond to specific fraud scenarios.[BPD ₀₃]	95
4.3	ELO Data: Impact of Privacy Budget on F1 Score Across Various Graph Pooling Methods.	99
4.4	Synthetic Data: Impact of Privacy Budget on F1 Score Across Various Graph Pooling Methods.	100
4.5	Temporal Embedding Distance	105
4.6	Spatial Embedding Distance	106
4.7	Bearing Embedding Distance	106
5.1	Automated Pipeline for Privacy-aware Vocabulary Corpus Builder.	117
5.2	Parallel Coordinates Analysis of 560 Hyperparameter Combinations Affecting Federated Tokenizer Performance: Fertility and Proportion of Continued Words. Optimal parameters are highlighted with a red line.	127
5.3	Parity Plot.	134

List of Tables

2.1	Comparison of Privacy Preservation Strategies in FL.	30
2.2	Comparison of Tokenization Methods in Language Models	48
3.1	Mathematical Symbols and Definitions	58
3.2	Detailed overview of graph-based methods categorized by their approach and core features, including graph-level and heterogeneity, with a focus on graph representations.	72
3.3	Set of business questions as metapaths by financial experts, where ● Merchant, ● Location, and ● Time stand for node types.	73
3.4	Performance of various graph-based methods in predicting cardholder attributes. Superscripts denote model compatibility: [‡] for heterogeneous graphs, [†] for homogeneous graphs, for node-level, and [§] for graph-level downstream tasks. Statistical significance is denoted as: ^{**} $p < 0.01$, [*] $p < 0.05$. . .	74
4.1	Performance of Different Graph Pooling Methods for Federated Anomaly Detection (Including Centralized Model).	101
4.2	Impact of Federated Anomaly Score (AS) Integration on Federated Fraud Detection Performance, Measured by F1 Score Improvements.	103
4.3	Correlation Analysis of Location and Temporal Vectorial Representations. The table summarizes the preservation of distance, orientation, and time relationships using different correlation methods.	104
5.1	Hyperparameter Tuning Space for the FedByteLevelBPE Algorithm	126
A.1	Comparative Analysis of Tokenizer Fertility Improvement (ϕ) at the Document Level Across Local, Pre-trained, and Federated Approaches	143
A.2	Comparative Analysis of Tokenizer Fertility Improvement (ϕ) at the Vocabulary Level Across Local, Pre-trained, and Federated Approaches	144

A.3	Comparative Analysis of Tokenizer Proportion of Continued Words Improvement (ψ) at the Document Level Across Local, Pre-trained, and Federated Approaches	145
A.4	Comparative Analysis of Tokenizer Proportion of Continued Words Improvement (ψ) at the Vocabulary Level Across Local, Pre-trained, and Federated Approaches	146

1

Introduction

CHAPTER CONTENTS

1.1	Context	16
1.2	Contributions and Organization	19
1.3	Dissemination	23

1.1 CONTEXT

In the age of data, the success of machine learning models has traditionally relied on the availability of large centralized datasets, which facilitate robust training and generalization. However, in practice, data is often fragmented across different organizations, each bound by strict privacy regulations that prevent the sharing or centralization of this valuable resource. This issue is particularly acute in the financial sector, where institutions manage high amounts of sensitive data subject to rigorous data protection laws such as the GDPR [Eur16].

For financial institutions, centralized access to pooled data is often restricted due to competitive, regulatory, ethical considerations, as well as technical concerns, as it creates a single point of failure, compromising the entire data set if breached. To give one example, in September 2017, data breach occurs at Equifax impacting 147 million customers subject to Personally Identifiable Information (PII). The financial institution was fined \$700 million for the breach [Fed19].

While companies with access to larger datasets gain a broader and more accurate view of sector-specific business, often referred to as the 'Bank Pooling' strategy [Eur17], large financial institutions have a strong competitive incentive to keep their data within their own boundaries to maintain a competitive edge.

However, in a regulatory compliance environment, collaborative strategies often lead to a cooperation-competition trade-off, leading institutions to prioritize their own data assets over broader collaborative efforts [Ana16]. Unfortunately, isolated datasets are often insufficient in size and diversity to train high-performing machine learning models capable of effectively addressing sector-specific downstream tasks, as they can inherit data-driven biases, as demonstrated by [BN21], where local trained credit score models showed a bias of 5-10% against minority and low-income groups.

Due to these factors, Federated Learning (FL) emerges as a promising collaborative strategy to effectively mitigate the cooperation-competition trade-off in strictly regulated environments. The motivation for institutions to participate in FL can stem from regulatory compliance or incentives. As introduced in [MRTZ17], FL is a distributed machine learning technique

that allows models to learn from the institution’s local datasets separately without the need for data centralization.

Although a key theoretical requirement of FL as a collaborative strategy is the use of a shared model across all participants to increase transparency and benefit from the collective learning process, this ideal is often not easily realized in practice. A recent CSSF^{*} thematic review on the use of AI in the financial sector highlights that institutions often develop and deploy different model families tailored to their specific sectoral needs, even when addressing the same problem [Com23]. This divergence significantly impacts the adoption of FL. According to the report, only 7% of the institutions have adopted FL, in contrast to 62% relying on centralized learning, 20% on reinforcement learning and 11% on transfer learning.

Even when institutions agree on a shared model, significant risks persist in FL [LXW22]. The exchange of model parameters in FL could introduce vulnerabilities, as attacks such as model inversion and membership inference can infer raw data from model access [KCMW24, HYC⁺22]. Although privacy mechanisms like differential privacy [KOV15, DRV10] and k-anonymity [SS98, Sweo2] are commonly employed to mitigate these risks, they may not provide complete protection. Given the iterative nature of model updates in FL, data can still be exposed. Recent studies have demonstrated that data reconstruction is feasible during training iterations across various data types, including textual data [MSDCS19, BDJV22], tabular data [VBDV23], and computer vision datasets [ZMB20, ZLH19, YMV⁺21]. These findings highlight the need for more robust privacy-preserving techniques, particularly for the input data used in FL. It is important to highlight that this is not a failure specific to FL, but rather a natural extension of research on attacks against centralized machine learning models [SSSS17, YZCL19, ZJP⁺20, FJR15, FLJ⁺14].

These issues underscore several critical challenges that must be addressed to realize a FL for the financial sector.

- **C1.** Diverse model development practices across institutions—whether in-house, out-

^{*}CSSF stands for Commission de Surveillance du Secteur Financier, the financial regulatory authority in Luxembourg.

sourced, or platform-based—pose significant challenges for integration within an FL system. This dissertation proposes strategies to overcome these challenges.

- **C2.** The performance of FL models heavily relies on the quantity, diversity, and quality of data across participants. However, this dependency does not uniformly apply to all sector-specific applications, particularly those with specialized data.
- **C3.** Current privacy research in FL focuses on securing gradient communication and aggregation, yet this may still risk compromising local datasets with personally identifiable information (PII). This dissertation introduces novel privacy mechanisms tailored for the financial sector.
- **C4.** FL research in fraud detection primarily depends on labeled data, with limited exploration of unlabeled or semi-labeled learning approaches for detecting transaction fraud and anomalies. This dissertation seeks to address this gap.

Our work in this dissertation takes steps toward addressing the challenges outlined above.

- Addressing **C1**: For transactional data, we implement a model stacking strategy, integrating Graph Neural Networks (GNNs) based federated anomaly scores into a centralized fraud detection model, thereby eliminating the need for a shared model among participants. Additionally, in the context of financial textual data, we explore a novel vocabulary adjustment strategy using a federated learning tokenizer. This approach enhances the language models used by different federation participants without requiring model to be shared among participants.
- Addressing **C2**: We develop a federated learning model for building language model tokenizers tailored. Our approach demonstrates that improvement gains are independent of data dimensionality or quality, primarily due to the diverse writing patterns within different financial sub-sectors. Similarly, training a federated anomaly detection model with negative sampling technique, as discussed in C4, show a quasiuniform performance improvements across all participants.

- Addressing **C3**: We introduce novel privacy mechanisms to handle personally identifiable information (PII). First as a modeling mechanism, by transforming PII data into an ego-centric static and dynamic graph, eliminating the need for unique identifiers in the FL model. Recognizing that graph-based models can still pose privacy risks due to sensitive attributes like location and timestamp, we extend Personalized Local Differential Privacy (PLDP) to introduce a personalized noising mechanism for temporal and spatial attributes. This ensures that even in the event of a data breach, the raw data remains private while still providing useful feature representations for the FL model using our proposed Privacy-Preserving Behavioral Anomaly Detection framework.
- Addressing **C4**: We incorporate domain-specific graph-based negative sampling techniques that automatically generate contextually relevant labels within dynamic graphs for training anomaly detectors across the federation. This approach enhances fraud detection accuracy with minimal alignment efforts required from federation participants.

Throughout this dissertation, we utilize various real-world datasets to test and validate our methodologies across different contexts. For ego-centric static graph modeling, we employed a private dataset provided by our industrial partner. In the development of federated dynamic graph anomaly and fraud detection, as well as the domain-specific graph-based negative sampling techniques, we used two publicly available real-world credit card datasets. Additionally, for financial textual analysis, we conducted downstream task testing using publicly available real-world complaint communication data collected from 30 financial institutions from the United States.

1.2 CONTRIBUTIONS AND ORGANIZATION

- **Chapter 2. Concepts & Related Work** presents the foundational concepts that support the methods and experiments conducted in this thesis. The chapter is divided into three main sections. The first section introduces **Federated Learning**, detailing

the fundamental principles and privacy mechanisms essential for decentralized learning. The second section explores **Graph Machine Learning**, covering key concepts such as Graph Neural Networks and their applications in different learning tasks. Finally, the chapter covers **Language Modeling**, providing an overview of foundational architectures with a particular emphasis on tokenization techniques and their role in processing and understanding textual data.

- **Chapter 3. Unsupervised Representation Learning for Heterogeneous Graphs in Credit Card Transactions** In this chapter, we present G-HIN2Vec [DSHS23a], a novel neural network architecture specifically designed for embedding entire heterogeneous information networks (HINs). Our approach leverages recent advancements in unsupervised learning techniques inspired by natural language processing (NLP), incorporating negative sampling methods to learn graph-level embeddings. G-HIN2Vec is applied to real-world credit card transaction data and benchmarked across various downstream tasks, including graph-level regression, classification, and clustering, against existing methods such as HIN2Vec [FLL17], DiffPool [YYM⁺18], and Metapath2Vec [DCS17]. Our experimental results demonstrate that G-HIN2Vec customer representation outperforms these baselines, achieving a 2.45% improvement in gender classification accuracy and a 7.19% increase in the R-squared (R^2) for income prediction. Additionally, our approach shows a 6.55% reduction in mean absolute error (MAE) for age prediction compared to the DiffPool method. These results highlight the effectiveness of G-HIN2Vec in representing customer unique identifiers within a latent space.

Concretely, our contributions in this chapter are as follows:

- We propose an ego-centric graph model as a mechanism to build an unlabeled heterogeneous graph dataset for cardholder transactions, inspired by egocentric thinking.
- We introduce G-HIN2Vec, a novel unsupervised representation learning method that employs a double-triplet loss function, enabling task-agnostic embeddings for entire heterogeneous graphs.

- We conduct an experimental evaluation, benchmarking G-HIN₂Vec against various GNN models on a real-world credit card private dataset.

- **Chapter 4. Federated Learning Framework for Privacy-Preserving Anomaly Detection in Financial Transactions** This chapter presents a novel federated learning (FL) framework designed for Privacy-Preserving Behavioral Anomaly Detection in the context of financial transactions. By leveraging Graph Neural Networks (GNNs) to model transactions as ego-centric dynamic homogeneous graphs, our approach shifts the focus from static to dynamic representations of customer behavior patterns, allowing for more accurate anomaly detection. A key innovation in our framework is the introduction of a domain-specific negative sampling technique that trains anomaly detection models without the need for labeled data, addressing the common challenge of scarce datasets in real-world applications. GNN-based methods in our framework achieve average F1-scores of 0.91 and 0.87 on two datasets, significantly outperforming traditional clustering-based methods. Additionally, by implementing a stacking strategy that uses the federated anomaly score as an input in both centralized and federated fraud detection systems, our framework improves F1-scores by an average of 2.1% and 1.76% over conventional supervised federated models. These results underscore the effectiveness of our approach in enhancing fraud detection while maintaining privacy.

Our contributions in this chapter are outlined as follows:

- We develop an end-to-end federated learning framework specifically designed for privacy-preserving behavioral anomaly detection in credit card transactions using dynamic graphs.
- We introduce a robust privacy-preserving feature engineering process that transforms spatial and temporal financial attributes into an embedding space, as a noise-based privacy mechanism, ensuring that individual privacy is protected while retaining key transactional characteristics.
- We propose a domain-specific negative sampling strategy that enriches the anomaly detection process by generating contextually relevant negative samples within

the dynamic graph framework.

- We validate the proposed framework through comprehensive evaluations on two benchmark datasets, demonstrating its superior performance in improving fraud detection while ensuring privacy protection.

- **Chapter 5. Federated Byte-level BPE Tokenizer: A Privacy-Preserving Approach for Distributed Language Tokenization** In this chapter, we present the Federated Byte-level Byte-Pair Encoding (FedByteBPE) Tokenizer, a novel approach that leverages Federated Learning (FL) to train Byte-level BPE tokenizers [WCG20] across distributed datasets while ensuring individuals and participants privacy preservation. Unlike traditional centralized methods, FedByteBPE allows institutions to train and refine their tokenizer models locally, with vocabulary aggregation performed on a centralized server, creating a robust, domain-specific tokenizer without compromising privacy. Our method is supported by both theoretical analysis and empirical evaluations on a public real-world distributed financial dataset. The results demonstrate that the federated tokenizer significantly outperforms off-the-shelf and locally trained tokenizers in terms of vocabulary coverage, showcasing the potential of federated learning for training tokenizers in privacy-preserving settings.

Concretely, our contributions in this chapter are as follows:

- We develop and introduce a Federated Byte-level BPE Tokenizer algorithm, designed to train tokenizers across distributed datasets in real-world scenarios while preserving privacy.
- We propose a novel vocabulary adjustment strategy that leverages the federated learning approach to enhance language models used by different federation participants without requiring model sharing, ensuring data confidentiality.
- We conduct a comparative analysis of state-of-the-art generic and domain-specific pretrained tokenizers, demonstrating that our federated approach not only competes with but also outperforms centralized pretrained tokenizers.
- We demonstrate that the improvement gains of our approach are independent of the data’s dimensionality or quality existing in different financial institutions.

1.3 DISSEMINATION

Our research has resulted in the following publications, presented in chronological order:

International conferences:

- **P1.** Damoun, F., Seba, H., Hilger, J., and State, R. "G-HIN₂Vec: Distributed heterogeneous graph representations for cardholder transactions." Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing. 2023. [DSHS23a]
- **P2.** Damoun, F., Seba, and State, R. "Federated Learning-Based Tokenizer for Domain-Specific Language Models in Finance." Proceedings of the International Conference on Advances in Social Networks Analysis and Mining, 2024. [DSS24a]
- **P3.** Damoun, F., Seba, and State, R. "Privacy-Preserving Behavioral Anomaly Detection in Dynamic Graphs for Card Transactions". Proceedings of the International Conference on Web Information Systems Engineering, 2024. [DSS24b].

International journals:

- **P4.** Damoun, F., Seba, H., Hilger, J., and State, R. "Graph-Level Heterogeneous Information Network Embeddings for Cardholder Transaction Analysis." Neural Computing and Applications (2024). [DSHS23b] (*Awaiting Editor-in-Chief Decision*)

The content of these publications forms the foundation of several chapters of this dissertation. Specifically:

- **P1** and **P4** are the basis for Chapter 3, which introduces G-HIN₂Vec, ego-centric graph modeling, and benchmarking on a private real-world dataset.
- **P3** forms the basis for Chapter 4, presenting the privacy-preserving behavioral anomaly detection framework, evaluated on both synthetic and real-world public datasets.

- **P2** forms the basis of Chapter 5, introducing the federated Byte-level Byte-Pair Encoding tokenizer, benchmarked against various language models on public dataset.

Chapters 4 and 5 extend the work presented in **P3** and **P2**, respectively.

Part I

Background

2

Concepts & Related Work

CHAPTER CONTENTS

2.1	Federated Learning Fundamentals	27
2.2	Graph Machine Learning	32
2.3	Language Modeling	41

2.1 FEDERATED LEARNING FUNDAMENTALS

Federated Learning was introduced in 2016 [MRTZ17] as a decentralized machine learning framework to enable multiple entities to collaboratively train a global shared model without sharing their local dataset with a central server [MBo8]. This approach addresses the increasing concerns over data privacy by keeping data local, while only exchanging model updates between participating entities. These updates are aggregated to form a global model, which progressively improves as the learning process continues.

In FL, each client k holds a local dataset D_k , which it uses to compute an updated version of the model parameters w . These updates are then aggregated by a central server to form a global model w_t , without directly accessing the local datasets. The global model is iteratively refined through rounds of communication between the clients and the server.

Mathematically, the objective is formalized as a minimization problem where the goal is to minimize the global objective function $F(w)$, defined as:

$$\min_w F(w) = \min_w \sum_{k=1}^K p_k F_k(w)$$

Here,

$$F_k(w) = \frac{1}{n_k} \sum_{j=1}^{n_k} f_{jk}(w; x_{jk}, y_{jk})$$

represents the local objective function for the k -th client, n_k is the number of data points at the k -th client, and $p_k = \frac{n_k}{n}$ denotes the relative contribution of each client, where $n = \sum_k n_k$ is the total number of data samples across all clients.

2.1.1 FEDERATED LEARNING ALGORITHMS

Aggregation algorithms are central to the FL process and have attracted significant research interest due to the unique challenges inherent in this learning paradigm. These algorithms are specifically designed to enhance communication efficiency, improve scalability, and adapted for non-independent and identically distributed data across clients.

The first aggregation algorithm introduced in FL is Federated Averaging (**FedAvg**) [MMR⁺17]. In **FedAvg**, each client k updates the global model by minimizing a local objective function $F_k(w)$ using its dataset D_k . The local model updates are computed using stochastic gradient descent (SGD):

$$w_{k,t+1} \leftarrow w_{k,t} - \eta \nabla F_k(w_{k,t}),$$

where η is the learning rate. The central server then aggregates these local updates to form the updated global model:

$$w_{t+1} = \sum_{k=1}^K p_k w_{k,t+1},$$

where $p_k = \frac{n_k}{n}$ represents the relative contribution of each client.

While **FedAvg** is effective in many situations, it encounters challenges when dealing with non-IID data distributions [KMAS⁺19]. To address these issues, the Federated Proximal (**FedProx**) algorithm was introduced, which modifies the local objective function by adding a proximal term:

$$F_k(w) = \frac{1}{n_k} \sum_{j=1}^{n_k} f(w; x_j, y_j) + \frac{\mu}{2} \|w - w_t\|^2,$$

where μ is a regularization parameter that controls the influence of the proximal term [LSZ⁺20].

This adjustment helps mitigate divergence in local models, especially in heterogeneous environments.

In addition to **FedAvg** and **FedProx**, other algorithms have been proposed, such as Federated Matched Averaging (**FedMA**) [WYS⁺20] addresses model architecture mismatches across clients, **FedNova** [WLLJ20] introduces normalized averaging to handle varying amounts of local updates, and **FedBN** [LJZ⁺21] is designed to improve performance in non-IID settings by using client-specific batch normalization layers.

2.1.2 PRIVACY MECHANISMS IN FEDERATED LEARNING

While FL offers significant advantages in preserving data privacy by keeping raw data on local devices, it is not completely immune to privacy risks. The model parameters exchanged between clients and the central server can still reveal sensitive information. Recent studies have demonstrated the effectiveness of various attacks, such as model inversion [FJR15] and membership inference [SSS17]. Other types of attacks can occur at different steps of the FL process, we categorize as follows:

- **Pre-Training:** Before feeding the data to the shared model, malicious clients may perform data poisoning or altering training labels to diverge the global model performance [SSS17, LWSV16].
- **During Training:** Over FL rounds, adversaries, could be a client or third-party, can introduce model poisoning by uploading random parameter updates, leading to reduced global model performance [XHCL19, SX18].
- **Post-Training:** After model final round, the global model is exposed to inference attacks that can be used to extract sensitive information from the model, such as determining if specific records were included in the training data [FJR15, MSDCS19].

To address these privacy risks, various families of mechanisms have been introduced in FL, including cryptographic-based, hardware-based, anonymization-based, and noise-based methods, as summarized in Table 2.1.

Table 2.1: Comparison of Privacy Preservation Strategies in FL.

Privacy Method	Approach	Advantages	Limitations	Complexity
Cryptographic-based	Encrypts data or computations using techniques like homomorphic encryption[Gen09] or secure multi-party computation (SMC)[BIK ⁺ 17].	<ul style="list-style-type: none"> - Strong confidentiality. - Supports secure collaborative computation without revealing data. 	<ul style="list-style-type: none"> - High computational requirements. - Slows down processing. - Difficult to manage. 	Very High
Hardware-based	Uses secure hardware components like Trusted Execution Environments such as Intel SGX processors [CD16] to protect data during processing.	<ul style="list-style-type: none"> - Strong protection during execution. - Minimal impact on data utility. 	<ul style="list-style-type: none"> - High cost. - Potential hardware vulnerabilities. - Complex deployment. 	High
Anonymization-based	Removes or generalizes identifiable information such as PII before data processing or sharing, common method is k-Anonymity [Swe02, SS98].	<ul style="list-style-type: none"> - Reduces risks of direct identification. - Suitable for high-dimensional datasets. 	<ul style="list-style-type: none"> - Re-identification risks through cross referenced quasi-identifiers. - Loss of data utility due to generalization. 	Moderate
Noise-based	Adds random noise to data or models to protect individual privacy while allowing data analysis, fundamentally proofed method is DP.	<ul style="list-style-type: none"> - Provides quantifiable privacy guarantees. - Customizable based on privacy needs. 	<ul style="list-style-type: none"> - Balancing privacy and accuracy is challenging. - Reduces data utility. 	Low

Among these, noise-based methods are particularly prominent, offering an privacy-utility trade-off and quantifiable risk. The most commonly employed techniques include Differential Privacy (DP), Local Differential Privacy (LDP), and Personalized Local Differential Privacy (PLDP). Below, we focus on these approaches.

DIFFERENTIAL PRIVACY (DP)

Differential Privacy (DP) [DR⁺14] is a formal framework designed to protect individual data privacy by ensuring that the output of a computation remains approximately the same, regardless of whether any single individual's data is included in the dataset. The privacy guarantee is characterized by two parameters: ϵ , which controls the privacy-utility trade-off, and δ , which represents the probability that the privacy guarantee might not hold. Formally, a randomized algorithm \mathcal{M} satisfies (ϵ, δ) -differential privacy if for any two datasets D and D' that differ in at most one element, and for any subset of outputs $O \subseteq \text{Range}(\mathcal{M})$, the following holds:

$$\Pr[\mathcal{M}(D) \in O] \leq e^\epsilon \Pr[\mathcal{M}(D') \in O] + \delta$$

Here, ϵ provides the upper bound on how much more likely an output is if an individual's data is included, while δ is the probability that the privacy guarantee may not hold. A smaller ϵ and δ signify stronger privacy guarantees, though at the potential cost of reduced accuracy [DR⁺14].

LOCAL DIFFERENTIAL PRIVACY (LDP)

Local Differential Privacy (LDP) [DJW13] extends the principles of DP to individual data points before they are sent to a central server. Under LDP, each client perturbs its data locally, ensuring that the server cannot directly access the original data. The (ϵ, δ) -LDP definition incorporates the δ parameter as follows:

$$\Pr[\mathcal{M}(v) = O] \leq e^\epsilon \Pr[\mathcal{M}(v') = O] + \delta$$

Here, \mathcal{M} is a randomized algorithm, and v and v' are two different inputs. This definition ensures that the output of \mathcal{M} does not reveal much about the individual input values, with δ capturing the probability of a privacy breach. This makes LDP particularly effective in scenarios where the central server is not fully trusted [KLN⁺11].

PERSONALIZED LOCAL DIFFERENTIAL PRIVACY (PLDP)

Personalized Local Differential Privacy (PLDP) [LMW⁺20] adapts the concept of LDP by allowing different clients to have varying levels of privacy based on their data sensitivity. This flexibility is achieved by allowing each client to define its own privacy budget ϵ_i and associated δ_i , which governs the amount of noise added to their data. The (τ, ϵ, δ) -PLDP is formally defined as:

$$\Pr(\mathcal{M}(v) \in O) \leq e^{\varepsilon_{\text{total}}} \Pr(\mathcal{M}(v') \in O) + \delta,$$

$$\text{where } \varepsilon_{\text{total}} = \varepsilon_1 + \varepsilon_2 + \dots + \varepsilon_n = \varepsilon'_1 + \varepsilon'_2 + \dots + \varepsilon'_n.$$

Here τ denotes the set of all possible inputs, where each ε_i (or ε'_i) is the personalized local privacy budget for each client i , and δ is the probability of a privacy failure. Such personalized local differential privacy provides more nuanced protection, letting each client control their own level of privacy [GBD17].

IMPORTANT PROPERTIES OF DIFFERENTIAL PRIVACY

Differential Privacy mechanisms are characterized by several key properties, including:

- **Post-processing Invariance:** If a mechanism \mathcal{M}_1 satisfies (ε, δ) -DP, then applying any randomized algorithm \mathcal{M}_2 to \mathcal{M}_1 still ensures that the result satisfies (ε, δ) -DP.
- **Sequential Composition:** When multiple mechanisms \mathcal{M}_i are applied sequentially on the same dataset, where each mechanism satisfies $(\varepsilon_i, \delta_i)$ -DP, the overall mechanism satisfies $\sum_i \varepsilon_i$ -DP with an accumulated δ .

These properties are crucial in ensuring that privacy guarantees are maintained even when multiple DP mechanisms are combined or when they are applied in sequence.

2.2 GRAPH MACHINE LEARNING

In this section, we present essential notions and definitions related to graph, graph neural networks and graph representation problem.

Graph machine learning is fundamentally machine learning based on graph data structure. These structures effectively model relationships between entities across various contexts. In

graphs, entities are represented as nodes, while their connections are represented as edges, offering a non-euclidean space to capture various interactions. This modeling approach is especially useful in different domains such as financial systems [KRS⁺₁₉, CWH₁₈, YWCW₁₉, MST₁₉], where it enables the analysis of both individual components and the global components in network structure. Given the inherent complexity of graph data, graph-based methods are introduced as robust tools for identifying patterns and detecting anomalies in different types of networks [MZM₂₃, LE₂₄].

2.2.1 PRELIMINARY NOTIONS

Definition 1 (Graph). *Formally, a graph is defined as a single fixed graph with a set of ordered pair $G = (\mathcal{V}, \mathcal{E})$, where \mathcal{V} is the set of vertices (or nodes) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. The vertex labeling function is $f_{\mathcal{V}} : \mathcal{V} \rightarrow \Sigma_{\mathcal{V}}$, and the edge labeling function is $f_{\mathcal{E}} : \mathcal{E} \rightarrow \Sigma_{\mathcal{E}}$. The graph is associated with a vertex labeling function $f_{\mathcal{V}} : \mathcal{V} \rightarrow \Sigma_{\mathcal{V}}$ and an edge labeling function $f_{\mathcal{E}} : \mathcal{E} \rightarrow \Sigma_{\mathcal{E}}$. The sets $\Sigma_{\mathcal{V}}$ and $\Sigma_{\mathcal{E}}$ represent the labels for nodes and edges, respectively.*

Definition 2 (Dynamic Graph). *A dynamic graph is defined as an evolving graph over time, where nodes and attributes can be added or deleted from the graph. At each time point t , the graph is represented as $G_t = (\mathcal{V}_t, \mathcal{E}_t)$, where \mathcal{V}_t and \mathcal{E}_t denote the sets of nodes and edges at time t . The sequence of dynamic graphs is $G = \{G_{t_1}, G_{t_2}, \dots, G_{t_N}\}$ over the time period t_1 to t_N .*

Definition 3 (Weighted Graph). *Despite static or dynamic context, a weighted graph is a graph where edges are assigned numerical values called weights, where each edge $e_{ij} \in \mathcal{E}$ has an associated weight $w_{ij} \in \mathbb{R}$.*

Definition 4 (Attributed Graph). *Despite static or dynamic context, an attributed graph is a graph where each node has an associated attributes, formally $f_{\mathcal{V}}(v_i) = X_i = (x_1(v_i), \dots, x_k(v_i))$, where X_i is a vector of k attributes.*

Definition 5 (Homogeneous Graph). *A homogeneous graph is any graph in which all nodes and edges share the same label. Formally, this satisfy k -ultrahomogeneous property, this means $|\Sigma_V| = |\Sigma_E| = 1$.*

Definition 6 (Heterogeneous Graph). *A heterogeneous graph is any graph where nodes and/or edges can have different labels. This is characterized by $|\Sigma_V| > 1$ and/or $|\Sigma_E| > 1$.*

Definition 7 (Self-Loop). *A self-loop is an edge that connects a node to itself. In any graph G , a self-loop is represented by an edge $e_{ii} \in \mathcal{E}$ connecting $v_i \in \mathcal{V}$ to itself.*

Definition 8 (Adjacency Matrix). *The adjacency matrix of a graph G , denoted A_G or for simplicity A , is a matrix where both rows and columns are indexed by identical orderings of the vertex set \mathcal{V}_G . The matrix is defined as:*

$$A[u, v] = \begin{cases} |\{e \in \mathcal{E} \mid e = (u, v)\}| & \text{if } u \neq v, \\ |\{e \in \mathcal{E} \mid e = (v, v)\}| & \text{if } u = v. \end{cases}$$

Definition 9 (Adjacent Vertices). *Two nodes in any graph are adjacent if they are connected by an edge. Formally, nodes v_i and v_j are adjacent if $e_{ij} \in \mathcal{E}$.*

Definition 10 (Adjacent Edges). *Two edges in any graph are adjacent if two edges share a node as in-degree or out-degree. Formally, edges e_{ij} and e_{jk} are adjacent if they both connected to node v_j .*

Definition 11 (Path Sequence). *A path from vertex v_0 to vertex v_n in any graph is an alternating sequence of nodes and/or edges, denoted in general as*

$$P = \langle v_0, e_1, v_1, e_2, \dots, v_{n-1}, e_n, v_n \rangle$$

Definition 12 (Path). *A path in a graph $G = (\mathcal{V}, \mathcal{E})$ is a sequence of vertices $p = (v_1, \dots, v_l)$ such that each consecutive pair (v_i, v_{i+1}) , where $1 \leq i < l$, is connected by an edge in \mathcal{E} .*

Definition 13 (Subgraph). *A graph $G' = (\mathcal{V}', \mathcal{E}')$ is a subgraph of a graph $G = (\mathcal{V}, \mathcal{E})$, denoted by $G' \subseteq G$, if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$.*

Definition 14 (Bipartite Graph). *A graph G is bipartite if its node set \mathcal{V} can be partitioned into two disjoint subsets \mathcal{V}_1 and \mathcal{V}_2 such that every edge $e \in \mathcal{E}$ connects a node in \mathcal{V}_1 to a node in \mathcal{V}_2 . Formally, this is a special case of a k -partite graph where $k = 2$.*

Definition 15 (Graph Dataset). *A graph dataset is a collection of graphs. Formally, a graph dataset is represented as $\mathcal{G} = \{G_1, G_2, \dots, G_N\}$, where each $G_i = (\mathcal{V}_i, \mathcal{E}_i)$ is a graph in the dataset, with \mathcal{V}_i and \mathcal{E}_i denoting the vertex and edge sets of the i -th graph. The dataset can consist of homogeneous or heterogeneous graphs, depending on the specific application.*

2.2.2 WHAT IS GRAPH LEARNING?

Graph machine learning extends the complexity of graph data modeling by integrating graph theory with advanced machine learning techniques. This combination allows for the extraction of meaningful data representations for various downstream tasks, including node-level, edge-level, and graph-level tasks. Each of these tasks requires the graph's components to be as numerical features, facilitating their use in downstream machine learning models, by formulating graph representation learning as a machine learning problem, where the structure and properties of the graph are used to generate embedding vectors.

The key types of graph representations in node, edge, and graph levels, are defined as follows:

Definition 16 (Node Embedding). *Let $G = (\mathcal{V}, \mathcal{E})$ be a graph, as defined in Definition [1]. Node embedding is a mapping function $f: v_i \rightarrow \mathbb{R}^d$ that encodes each node $v_i \in \mathcal{V}$ into a low-dimensional vector of dimension d , where $d \ll |\mathcal{V}|$, such that the similarities between nodes in the original graph are preserved in the embedding space.*

Definition 17 (Edge Embedding). *Let $G = (\mathcal{V}, \mathcal{E})$ be a graph, as defined in Definition [1]. Edge embedding is a mapping function $f: e_{ij} \rightarrow \mathbb{R}^d$ that encodes each edge $e_{ij} \in \mathcal{E}$ into a low-dimensional vector of dimension d , where $d \ll |\mathcal{V}|$, ensuring that the similarities between edges in the original graph are preserved in the embedding space.*

Definition 18 (Graph Embedding). *Let \mathcal{G} be a graph dataset of size N , as defined in Definition 15. Graph embedding is a method that maps each graph $G_i \in \mathcal{G}$ into a low-dimensional vector space of dimension d , while preserving the structural and relational properties of the graphs. This embedding ensures that both inter- and intra-graph similarities are maintained within the embedding space.*

In general, the process of mapping graph substructures to dense and low-dimensional vector space can involve simple descriptive statistics at the node, edge, or graph level as vectorial representations [SSVL⁺11], or a dimensionality reduction algorithm applied to graph attributes [BN03], or more advanced deep learning methods [GL16, PARS14, TQW⁺15]. These advanced methods iteratively learn from initial random representations to meaningful representations tailored to specific downstream tasks, optimizing an objective function during the learning process, or a mixture of both as in [YV15a, BDQ⁺19]. In this thesis, we focus on deep learning-based methods for graph representation learning.

Deep learning-based methods for graph representation can be categorized into traditional graph embedding methods and Graph Neural Networks (GNNs). Traditional methods include factorization-based approaches, random walk-based techniques, and non-GNN deep learning methods [GL16, PARS14, TQW⁺15]. For an extensive survey on non-GNN methods, we refer to [KA24]. Below, we introduce the fundamentals of GNN-based deep learning methods and their techniques discussed in this thesis.

2.2.3 GRAPH NEURAL NETWORKS

GNNs can be defined as a class of neural networks designed to operate on graph-structured data without the need for any proxy transformations, such as NLP-based methods [PARS14, GL16]. GNNs employ different techniques that differ from traditional methods, particularly in their ability to generalize to unseen nodes and leverage node and edge attributes more effectively. Simply, GNNs rely on node's representation (or embedding) from its neighbors within the graph topology, this allows to capture both structural features, with a flexible local-global structural trade-off depending on the specific downstream task.

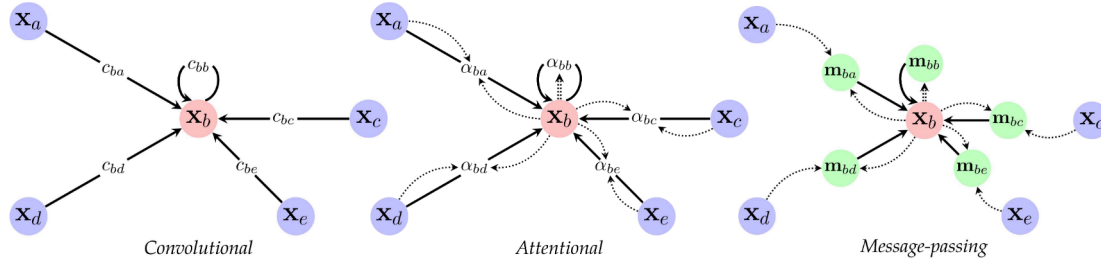


Figure 2.1: Illustration of graph convolution operations. (*)

The core operation in many GNN architectures is the graph convolution operation, initially introduced in [DBV16], which is used to generate node representation vectors. Given the features $\mathbf{h}_v^{(0)}$ for all nodes $v \in V$, the graph convolution operation, as introduced in Graph Convolutional Networks (GCNs) in [KW17, WSZ⁺19], can be expressed as:

$$\mathbf{h}_v^{(k+1)} = \sigma \left(\mathbf{h}_v^{(k)}, \bigoplus_{u \in \mathcal{N}(v)} w_{vu} \psi(\mathbf{h}_u^{(k)}) \right)$$

Here, $\mathbf{h}_v^{(k)}$ represents the embedding of node v at layer k , $\mathcal{N}(v)$ denotes the set of neighbors of node v , w_{vu} is a learned weight associated with the edge $e_{vu} \in E$ (or directly derived from the adjacency matrix if not learnable), ψ is a learnable function, and σ is an activation function, such as ReLU.

* Credit: Petar Veličković, DeepMind.

This operation act as an aggregation of from a node's neighbors information and applies a transformation to generate the next layer's node representation. This foundational approach is known for its computational expressiveness and has led to the development of various GNN models by defining different graph convolution operations:

- **Attentional-based Convolution Operation** [MBM⁺17, VCC⁺18, ZSX⁺18]:

$$\mathbf{h}_v^{(k+1)} = \sigma \left(\mathbf{h}_v^{(k)}, \bigoplus_{u \in \mathcal{N}(v)} a(\mathbf{h}_v^{(k)}, \mathbf{h}_u^{(k)}) \psi(\mathbf{h}_u^{(k)}) \right)$$

In this variant, $a(\mathbf{h}_v^{(k)}, \mathbf{h}_u^{(k)})$ represents attention weights, denoted as a_{vu} for the edge $e_{vu} \in E$.

- **Message-Passing-based Convolution Operation** [BPL⁺16, GSR⁺17, BHB⁺18]:

$$\mathbf{h}_v^{(k+1)} = \sigma \left(\mathbf{h}_v^{(k)}, \bigoplus_{u \in \mathcal{N}(v)} \psi(\mathbf{h}_v^{(k)}, \mathbf{h}_u^{(k)}) \right)$$

Here, $\psi(\mathbf{h}_v^{(k)}, \mathbf{h}_u^{(k)})$ represents the message-passing function, with the computed message denoted as m_{vu} for the edge $e_{vu} \in E$.

- **Sample-Aggregate Operation**[HYL17]:

$$\mathbf{h}_v^{(k+1)} = \sigma \left(\mathbf{W}_k \cdot \text{CONCAT} \left(\mathbf{h}_v^{(k)}, \text{AGGREGATE} \left(\{\mathbf{h}_u^{(k)}, \forall u \in \mathcal{N}(v)\} \right) \right) \right)$$

In this approach, the aggregation and concatenation operations are performed on randomly sampled nodes from the entire graph, rather than aggregating over all neighbors.

These various formulations highlight the flexibility of GNNs in adapting to different graph structures and computational constraints, offering diverse approaches to leverage the rich information present in graph data.

As the final layer in a GNN architecture provides node embedding representations through a feedforward neural network, the training process can operate in supervised, semi-supervised, self-supervised, or unsupervised frameworks, depending on the specific machine learning problem. Below, we describe the final layer(s) stacked after GNN layers to perform various downstream tasks:

1. Node-Level Downstream Task

For node-level tasks, the GNN layers output node representations $\mathbf{h}_v^{(K)}$ for each node v after K layers. The prediction output for node v can be generated using a multi-layer perceptron (MLP) or a softmax layer (for classification) or a regression layer:

- **Classification:**

$$\hat{y}_v = \text{softmax}(\mathbf{W}_{\text{out}}\mathbf{h}_v^{(K)} + \mathbf{b}_{\text{out}})$$

where \mathbf{W}_{out} and \mathbf{b}_{out} are the learnable weights and bias of the output layer, respectively.

- **Regression:**

$$\hat{y}_v = \text{MLP}(\mathbf{h}_v^{(K)})$$

where the MLP is a feedforward neural network that outputs the predicted value \hat{y}_v for node v .

2. Edge-Level Downstream Task

For edge-level tasks, the GNN layers output representations $\mathbf{h}_v^{(K)}$ and $\mathbf{h}_u^{(K)}$ for nodes v and u , respectively. A similarity function or an MLP can be used to predict the existence or weight of an edge between nodes v and u :

$$\hat{y}_{vu} = \sigma(\text{MLP}(\text{CONCAT}(\mathbf{h}_v^{(K)}, \mathbf{h}_u^{(K)})))$$

where σ is a sigmoid function used for binary classification task and CONCAT denotes the concatenation of the node embeddings.

Alternatively, a similarity function such as the dot product can be used:

$$\hat{y}_{vu} = \mathbf{h}_v^{(K)} \cdot \mathbf{h}_u^{(K)}$$

3. Graph-Level Downstream Task

For graph-level tasks, a graph representation \mathbf{h}_G is often generated by applying a readout layer on node representations $\mathbf{h}_v^{(K)}$ for all nodes v in the graph G . The readout function can be a pooling operation, which aggregates the node representations into a single graph representation vector:

- **Pooling Operation:**

$$\mathbf{h}_G = \text{POOL} \left(\{ \mathbf{h}_v^{(K)} \mid v \in G \} \right)$$

where POOL layer could be a mean, sum, or max pooling operation. Advanced methods like DiffPool [DBV16] introduce a differentiable pooling mechanism where nodes are assigned to clusters via a learned assignment matrix $\mathbf{S}^{(k)}$, which generates coarsened graphs for subsequent GNN layers. Other methods, such as ASAPool [MBM⁺17], leverage a self-attention mechanism combined with graph structure to adaptively select and pool the most informative nodes, allowing the network to retain both structural and feature-based information during pooling.

The graph representation \mathbf{h}_G can then be used for graph-level classification or regression tasks:

- **Graph Classification:**

$$\hat{y}_G = \text{softmax} (\mathbf{W}_{\text{out}} \mathbf{h}_G + \mathbf{b}_{\text{out}})$$

- **Graph Regression:**

$$\hat{y}_G = \text{MLP} (\mathbf{h}_G)$$

In this dissertation, we focus on graph-level downstream tasks using state-of-the-art GNN models. We perform extensive benchmarking across various GNN families to identify the

most suitable model for our specific downstream tasks, given the inherent constraints in federated learning.

2.3 LANGUAGE MODELING

In its broadest sense, Natural Language Processing (NLP) involves a wide range of tasks focused on the automatic processing of human language, such as text summarization, classification, and translation. Despite the diversity of these tasks, they all fundamentally depend on Language Models (LMs), which are complex neural network architectures designed to model natural language patterns effectively with contextual awareness, but not limited to textual data as an input and can be extended to other modalities where language plays a role.

Given the inherent complexity of natural languages, LM-based methods have proposed as powerful tools that rely on several key components working together: tokenization, embeddings, encoders, decoders components and related concepts such as pre-training and fine-tuning, where pre-trained LMs serve as repositories of transferable linguistic knowledge that can be fine-tuned for downstream tasks using smaller, task-specific datasets, allowing for efficient and accurate language models tailored to domain-specific tasks.

In this section, we define the basic definitions and essential notations of language modeling components investigated in this thesis, with a specific focus on tokenization component. Henceforth, all references to language modeling are defined here.

2.3.1 PRELIMINARY NOTIONS

Definition 19 (Language Model). *A language model is a probabilistic framework that estimates the likelihood of a sequence of words occurring in a given context [Sha51]. In the context of an ***N-gram model*** and given a sequence of words $\{w_1, w_2, \dots, w_n\}$, the model approximates*

the joint probability of the sequence using the chain rule of probability as follows:

$$Pr(w_1, w_2, \dots, w_n) \simeq \prod_{i=1}^n Pr(w_i | w_{i-N+1}, \dots, w_{i-1}),$$

where N is the size of the context window, and $Pr(w_i | w_{i-N+1}, \dots, w_{i-1})$ represents the probability of the word w_i occurring given the preceding $N - 1$ words, known as the context. As $N \rightarrow \infty$, the probability approximation of a word becomes more accurate by capturing longer dependencies between words.

Definition 20 (Neural Language Model). *A Neural Language Model is a probabilistic framework that uses (deep) neural networks to estimate the likelihood of a word sequence [BDV00, MKB⁺10]. In the context of an **N-gram model** and given a sequence of words $\{w_1, w_2, \dots, w_n\}$, the model approximates the joint probability as:*

$$Pr(w_1, w_2, \dots, w_n) \approx \prod_{i=1}^n Pr(w_i | w_{i-N+1}, \dots, w_{i-1}; \theta),$$

where Pr is estimated by a neural network with learned parameters θ , generalizing to unseen sequences while being constrained by a fixed-size context window N .

Definition 21 (Tokenization). *Tokenization is the process of converting a string, which can represent a word, sentence, or document, $S = \{s_1, s_2, \dots, s_n\}$, where s_i represents a sequence of individual characters or symbols, into a sequence of tokens $T = \{t_1, t_2, \dots, t_m\}$, where $n \geq m$ and each t_j can represent a word, subword, or character. Formally, given an input string S , the tokenization function $\mathcal{T} : S \rightarrow T$ generates the token sequence T according to predefined rules or algorithms:*

$$T = \mathcal{T}(S) = \{t_1, t_2, \dots, t_m\}$$

The number of tokens in T is determined by the learned vocabulary V by \mathcal{T} , and the token granularity depends on the chosen tokenization strategy, such as word-level, character-level, or subword-level methods.

Definition 22 (Text Encoding). *Text Encoding is the process of mapping discrete symbols from a set V , referred to as a pre-defined vocabulary, into a sparse vector space $\mathbb{R}^{|V|}$, where $|V|$ is the*

size of the vocabulary. Each symbol $v_i \in V$ is assigned a unique vector representation using an encoding function H .

In language models, the **initial text encoding**, also known as **one-hot encoding**, maps each symbol $v_i \in V$ to a binary vector using $H : V \rightarrow [0, 1]^{|V|}$, where:

$$H(v_i) = \bigoplus_{j=1}^{|V|} \delta_{ij}$$

Here, δ_{ij} is the Kronecker delta function, which takes the value 1 if $i = j$ and 0 otherwise. This ensures that each symbol is represented as a unique orthogonal vector in $[0, 1]^{|V|}$, with only one non-zero entry.

Definition 23 (Word Embeddings). *Word Embeddings are dense vector representations of words from a set V , referred to as a pre-defined vocabulary, mapped into a lower-dimensional vector space \mathbb{R}^d , where $|V| \gg d$. The embedding function $\mathbf{E} : V \rightarrow \mathbb{R}^d$ assigns each word $w_i \in V$ a "unique" vector $\mathbf{E}(w_i)$, within d -dimensional latent space with learned or static properties.*

Formally, the embedding layer is defined by an embedding matrix $E \in \mathbb{R}^{|V| \times d}$, which acts as a lookup table. The input word w_i , represented by its one-hot encoded vector $H(w_i) \in \mathbb{R}^{|V|}$, is mapped to its corresponding dense vector through a matrix multiplication:

$$\mathbf{E}(w_i) = H(w_i)^\top E = E[i] = [e_1, e_2, \dots, e_d] \in \mathbb{R}^d$$

Here, $E[i]$ represents the i -th row of the embedding matrix E , corresponding to the word w_i .

Definition 24 (Pre-training). *Pre-training is the process of training language models on a large, unlabeled corpus of documents to learn general representations. These learned representations can later be used to specific downstream tasks. Within a self-supervised paradigm, pre-training leverages the data itself to generate supervision signals, enabling the model to solve tasks such as predicting context words based on a target word, predicting the target word based on its context, predicting the next word given previous words, or masking a word in a sequence and predicting it from the surrounding words.*

Pre-training Details. Given a corpus $\mathcal{D}_{\text{pre}} = \{d_1, d_2, \dots, d_N\}$, where each document d_i contains tokenized words $\{w_1, w_2, \dots, w_n\}$ from the vocabulary V , the objective of pre-training is to predict parts of the text from other parts. This is commonly achieved through various self-supervised tasks:

- **Skip-gram [MCCD13]:** The task is to predict context words w_{i+j} given a target word w_i within a window size k . The objective function is:

$$\theta_{\text{pre}} = \arg \max_{\theta} \sum_{i=1}^T \sum_{-k \leq j \leq k, j \neq 0} \log \Pr(w_{i+j} \mid w_i; \theta)$$

- **CBOW (Continuous Bag of Words) [MCCD13]:** The task is to predict the target word w_i given the surrounding context words $\{w_{i-k}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+k}\}$. The objective function is:

$$\theta_{\text{pre}} = \arg \max_{\theta} \sum_{i=1}^T \log \Pr(w_i \mid w_{i-k}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+k}; \theta)$$

- **Next Token Prediction [RNSS18]:** In autoregressive models, the task is to predict the next token w_{i+1} in a sequence given all preceding tokens $\{w_1, w_2, \dots, w_i\}$. The objective function is:

$$\theta_{\text{pre}} = \arg \max_{\theta} \sum_{i=1}^T \log \Pr(w_{i+1} \mid w_1, w_2, \dots, w_i; \theta)$$

- **Masked Language Modeling (MLM) [DCLT19]:** In this task, random words in the sequence are masked, and the model predicts the original words given the surrounding context. For a masked word $w_{[\text{MASK}]}$ in a document, the objective is:

$$\theta_{\text{pre}} = \arg \max_{\theta} \sum_{i=1}^T \log \Pr(w_{[\text{MASK}]} \mid w_1, \dots, w_{[\text{MASK}]-1}, w_{[\text{MASK}]+1}, \dots, w_n; \theta)$$

In each of these tasks, $Pr(\cdot)$ is the probability model parametrized by the embedding matrix E and neural network parameters θ .

Definition 25 (Fine-tuning). *Fine-tuning is the process of adapting a pre-trained language model to a specific downstream task by training it on a task-specific labeled dataset. During fine-tuning, the model's parameters, which were initialized during pre-training, are updated to optimize the performance for a particular application with significantly less data than pre-training, as the model already has general language understanding and knowledge.*

Fine-tuning Details. Given a labeled dataset $\mathcal{D}_{\text{fine}} = \{(d_1, y_1), (d_2, y_2), \dots, (d_N, y_N)\}$, where each document d_i contains tokenized words $\{w_1, w_2, \dots, w_n\}$ and corresponding task-specific labels y_i , the objective of fine-tuning is to adjust the pre-trained model for a specific task. This is achieved by minimizing a task-specific loss function:

- For classification tasks, where the labels y_i are discrete classes, the fine-tuning process minimizes the **cross-entropy loss function**:

$$\theta_{\text{fine}} = \arg \min_{\theta} \sum_{i=1}^N \left[- \sum_{c=1}^C y_{i,c} \log Pr(c \mid d_i; \theta) \right]$$

where $Pr(c \mid d_i)$ is the predicted probability of class c for input d_i , and $y_{i,c}$ is the one-hot encoded true label for class c .

- For regression tasks, where the labels y_i are continuous values, the fine-tuning process minimizes the **mean squared error (MSE) loss function**:

$$\theta_{\text{fine}} = \arg \min_{\theta} \sum_{i=1}^N (\hat{y}_i - y_i)^2$$

where \hat{y}_i is the predicted value for input d_i , and y_i is the true continuous label.

In both cases, the fine-tuning process updates the pre-trained model's parameters θ_{pre} to adjust the model to the specific task, resulting in optimized parameters θ_{fine} .

2.3.2 WHAT IS LANGUAGE MODELING ?

Language models are systems designed to understand and generate natural language by predicting the probability of a word or sequence of words within a given context by capturing textual patterns using contextual representations to better model linguistic structures. At the time of writing, **GPT (Generative Pre-trained Transformer)** is the state-of-the-art language model, using the *Transformer architecture* [VSP⁺17]. GPT models are autoregressive within a self-supervised paradigm, meaning they predict text one word at a time based on previous words, which makes them highly effective for a wide range of downstream tasks.

From an evolution perspective, the language model core module is to estimate the probability distribution of a word sequence [Sha51]. Earlier models, such as N-gram models [BDV00, MKB⁺10], relied on fixed-size windows to predict the likelihood of the next word based on a limited context of preceding words. While effective for capturing short-range dependencies, these models struggled with larger vocabularies and failed to model long-range dependencies effectively. As the vocabulary and context size increase, the performance of these models degraded due to the dimensionality and distant words in a sequence of words.

To address these limitations, neural language models (see Definition 20) such as Recurrent Neural Networks (RNNs) [CGCB14] and later Long Short-Term Memory (LSTM) [Hoc97] networks were introduced. These architectures incorporated memory components that allowed the language model to maintain a contextual representation of previous words over longer sequences beyond the fixed windows used by N-gram models. However, despite these advancements, the performance of these models degraded due to very long sequences to retain information over extended text spans [JVS⁺16].

In recent years, the introduction of the Transformer architecture [VSP⁺17] overcame these limitations departed from the sequential nature of RNNs by leveraging a self-attention mechanism, which allows the model to weigh the importance of all words in a sequence, enabling to capture both short- and long- range dependencies without the sequential processing and fixed windows constraints, resulting in significant improvements in both text understanding and generation tasks. Transformer-based language models are typically categorized into

autoregressive models and masked language models (MLM).

- **Autoregressive Models:** Models like GPT [RNSS18] generate the next word in a sequence based on preceding words, using a left-to-right context. Each new token is conditioned on the previously generated sequence.
- **Masked Language Models (MLM):** Models like BERT [DCLT19] mask certain tokens in the input and predict the masked tokens using context from both directions. This bidirectional approach enables a deeper contextual understanding.

They can also be implemented using different architectures: decoder-only, encoder-only, and encoder-decoder. In decoder-only models only the decoder component is used to generate output sequentially. Encoder-only models focus solely on the encoder to process the entire input sequence simultaneously. Encoder-decoder models, often used in sequence-to-sequence tasks like machine translation, involve both components, with the encoder processing the input and the decoder generating the output. For a detailed review, we refer to [WLW⁺23].

2.3.3 TOKENIZATION IN LANGUAGE MODELS

Tokenization (see Definition 2.1) is the first step and the hardest in language models where it convert raw text into units such as characters, words, subwords, or symbols, allowing the model to process input more effectively. Effective tokenization facilitates the handling of text with various structures, lengths, and complexities, improving the model's ability to learn better contextual representation in natural language.

Table 2.2: Comparison of Tokenization Methods in Language Models

Tokenization Method	Approach	Advantages	Limitations
Character-level	Splits text into individual characters [ZZL15].	<ul style="list-style-type: none"> - Very simple and language-agnostic. - Avoids OOV issues completely. 	<ul style="list-style-type: none"> - Longer sequences due to tokenizing each character. - Difficult to capture semantic meaning from individual characters.
Word-level	Splits text into words based on spaces and punctuation [NTPV20].	<ul style="list-style-type: none"> - Easy to implement. - Works well for languages with clear word boundaries. 	<ul style="list-style-type: none"> - Struggles with OOV words. - Inefficient for morphologically rich languages.
WordPiece	Merges subwords based on the most likely segments, using frequency and context [WCG20].	<ul style="list-style-type: none"> - Balances between frequency and context relevance. - Reduces OOV issues. 	<ul style="list-style-type: none"> - Slightly more complex than BPE. - Still suffers from fixed vocabulary issues.
SentencePiece	Tokenizes text at the sentence level without pre-tokenization [KR18].	<ul style="list-style-type: none"> - Effective for languages without clear word boundaries. - Avoids pre-tokenization issues. 	<ul style="list-style-type: none"> - Might segment sentences too aggressively in some cases. - Complex implementation.
Byte-Pair Encoding (BPE)	Iteratively merges the most frequent pairs of characters or subwords [SHB15].	<ul style="list-style-type: none"> - Handles OOV words effectively. - Efficient for morphologically complex languages. 	<ul style="list-style-type: none"> - Fixed vocabulary size. - May split meaningful words unnaturally.
Byte-level BPE	Extends BPE by processing text at the byte level [WCG20].	<ul style="list-style-type: none"> - Efficient for handling diverse scripts and character encodings. - Reduces tokenization errors in non-Latin scripts. 	<ul style="list-style-type: none"> - Can result in longer sequences due to byte-level splitting.

Early language models primarily relied on word-level tokenization [NTPV20], which splits text into words based on spaces and punctuation using pre-defined regular expressions. However, this approach is limited, particularly in morphologically rich languages and those without explicit word boundaries. Word-level tokenization also struggles with out-of-vocabulary (OOV) words, rare terms, and spelling variations, limiting its ability to generalize across diverse texts.

To address these issues, subword-level tokenization methods have become standard due to their improved generalization and flexibility. WordPiece [WCG20] operates by merging subword segments based on their contextual likelihood, while SentencePiece [KR18] adopts a slightly different strategy by operating at the sentence level without pre-tokenization step,

making it effective for languages with complex morphology.

Another widely adopted subword tokenization method is Byte-Pair Encoding (BPE) [SHB15] generalized by Byte-level BPE to cover any language by operating on byte codes [WCG20]. BPE begins by splitting text into individual characters or byte codes and progressively merges frequent byte code or character or subword pairs into larger tokens, drawing inspiration from text compression techniques. This method is particularly effective at handling OOV words, allowing models to break down rare or unseen words into meaningful subword units, thereby improving model performance across a variety of tasks.

Part II

Contributions

3

Unsupervised Representation Learning for Heterogeneous Graphs in Credit Card Transactions

CHAPTER CONTENTS

3.1	Introduction	53
3.2	Related Studies	55
3.3	Problem statement	57
3.4	Our approach	62
3.5	Design of experiments	70
3.6	Experiments and results	74
3.7	Conclusion	78

3.1 INTRODUCTION

Financial institutions manage a variety of transaction types, encompassing a spectrum of financial assets, such as loans, insurance, and investments. In today’s increasingly digital era, banks, as ubiquitous entities, offer several financial card products that dominate the retail banking sector. The past decade has seen an impressive increase in cashless payments through plastic and virtual cards, leading to exponential growth in the digital footprint of card transactions*.

Analyzing customer transactions is pivotal for various banking applications, including but not limited to behavior modeling, product recommendation, and advertising strategies [DCLOT⁺18]. Recent research in financial data analysis [RZZ⁺21, KRS⁺19] suggests the utility of graph data structures to encapsulate the topological information inherent in tabular data, thereby extending and enriching available customer data. Currently, deep learning algorithms have been refined and enhanced, expanding their applicability to a wider range of problems [WZT⁺23, SWS⁺23, SWSS23].

Despite the advantages, graph data structures introduce additional complexity, with each graph component characterized by unique attributes and local topology. Tasks such as graph classification, clustering, and regression, which require vectorial representations of graphs for the application of machine learning algorithms, become more challenging. Techniques to overcome this challenge include Graph Kernel algorithms [VSKB10], acting as hand-crafted feature extractors, and embedding algorithms that provide dense latent graph encoding [GF18].

However, most of these embedding techniques have been designed for homogeneous networks [GF18], and therefore their application is limited to graphs with a single type of nodes and edges. However, real-world networks often comprise heterogeneous information networks (HINs), where there are different types of interaction between graph components. Graph Neural Networks (GNNs) offer potential solutions for such graphs. GNNs have

*According to the Federal Prof. of Credit Card Operations of Depository Institutions report, 2020 ref.<https://www.federalreserve.gov/publications/files/ccprofit2020.pdf>

proven their effectiveness in various graph-related tasks, but they often fail to preserve the network structure fully, especially for HINs [WBS⁺22, WPC⁺20]. Graph analysis tasks are often confined by the characteristics of the problem at hand and the applied embedding techniques [CZC18]. In scenarios involving node-level tasks such as node regression and classification, the embeddings are typically designed to portray low-dimensional representations of nodes. For example, the classification of scientific publications using citation networks exemplifies a node-level task. Conversely, for graph-level tasks, such as graph classification and regression, the embeddings provide a low-dimensional representation of the entire graph. Examples of these tasks include prediction of protein functions using chemical compound graphs and detection of malware based on call graphs [MAWY21, LCY⁺18].

However, in the complexity of real-world scenarios, networks are typically heterogeneous information networks (HIN), comprising a multitude of interactions between different types of graph components. This heterogeneity brings a higher degree of complexity than that encountered in homogeneous networks, introducing diverse nodes and edges. In light of this, Graph Neural Networks (GNNs) have emerged as a subset of deep learning methods designed to handle graphs with auxiliary information. Despite their robustness and proven effectiveness in various graph-related tasks, GNNs often fall short in fully preserving the network structure in both homogeneous graphs (HG) and HIN [WBS⁺22, WPC⁺20], a critical aspect of graph analytics, especially for HINs.

Recently, random walk-based embedding methods have displayed significant performance improvements in various homogeneous graph (HG) downstream tasks by preserving the network structure [GF18]. However, their utility is considerably limited when applied to a heterogeneous schema. Concurrently, much of the existing research on graph embedding remains heavily focused on learning node representations, leaving a gap in the comprehensive understanding and handling of complex heterogeneous networks.

In this chapter, we introduce G-HIN2Vec specifically designed to embed entire heterogeneous graphs. It relies on joint learning architectures [SHH⁺18] and random walk-based methods for node embedding. We evaluate the quality of the generated heterogeneous graph

embeddings across various downstream tasks using different metrics for credit card transactions. Based on a preliminary literature review, we carefully selected our baseline methods. The experimental results show that G-HIN₂VEC embeddings outperform the baseline methods HIN₂VEC [FLL⁺17] and Metapath₂Vec [DCS⁺17]. In particular, G-HIN₂VEC exhibits a 2.45% improvement in the precision of the gender classification and a 7.19% increase in the R-squared income prediction (R^2) over the strongest baselines, with a loss in the Mean Absolute Error (MAE) in age prediction by 6.55% compared to DiffPool [YYM⁺18]. This shows the effectiveness of our approach in enhancing customer understanding in retail banking. Our contributions are as follows.

- We propose an ego-centric graph model for cardholder transactions to build an unlabeled heterogeneous graph dataset inspired by egocentric thinking.
- We introduce G-HIN₂Vec, a new unsupervised representation learning method that employs a double-triplet loss function as task-agnostic approach for entire heterogeneous graph representations.
- We experimentally benchmark various GNN models against G-HIN₂VEC on real-world credit card datasets for multiple downstream tasks, including graph classification, regression, and clustering tasks.

3.2 RELATED STUDIES

In this chapter, we review three primary research lines, focusing on heterogeneous information networks, graph-level embedding, and financial card transaction data analysis.

- **Heterogeneous Information Network (HIN) Embedding:** Heterogeneous Information Network Embedding has emerged as a significant area of research, primarily due to its ability to effectively represent complex real-world data. These networks encompass interactions among various types of objects, offering a more detailed representation of real-world scenarios.

Recent algorithms in HIN embedding primarily focus on node or edge-level embedding, as evidenced in studies by [DCS17, FLL17, WDL⁺19]. This focus, however, introduces constraints when addressing graph-level downstream tasks. In a recent comprehensive review [YXZ⁺20], 13 HIN embedding methods were benchmarked on four distinct datasets designed for node- and edge-level tasks. These studies have consistently demonstrated that the effectiveness of an HIN embedding algorithm depends on the specific requirements of the downstream task, underlying the graph schema, and other experimental variables, as highlighted in [WBS⁺22, WPC⁺20]. A critical limitation in the field is the lack of a unified framework for HIN embeddings, as pointed out by [GF18]. This absence has been identified as a significant barrier, particularly when addressing graph-level downstream tasks.

- **Graph-level Embedding:** Despite the abundance of research on node, edge, or sub-graph embedding algorithms, studies on entire graph embedding algorithms are relatively sparse [MAWY21]. This impacts the downstream graph-related tasks for HIN networks such as graph classification, regression, and clustering. Conversely, HG embeddings have been extensively studied and are well-established in various downstream tasks [NCV⁺17]. Both graph types embedding algorithms seek to preserve topological and semantic similarity across a graph dataset. As discussed in [MAWY21] and due to lack of HIN graph-level embedding algorithms, researchers often resort to transforming HINs into HG datasets. This transformation is a proxy approach to leverage existing homogeneous graph embedding algorithms, such as Graph2Vec [NCV⁺17] and Deep Graph Kernels (DGK) [YV15a], for generating graph-level embeddings. While this method has led to improvements in performance, it unfortunately discards potentially valuable semantic information. This limitation has pushed research towards developing embedding models tailored for HINs. UGraphEmb [BDQ⁺19] addresses the issue as a supervised training model that relies on existing heuristics for graph similarity [BK05, SVP⁺09], which is computationally intensive and not scalable. Instead of proxy or supervised approaches, HIN representation can be generated via aggregate functions, such as the average layer, to transform node/edge embeddings into graph embeddings, which often leads to suboptimal results [XHLJ18]. This is due to the

absence of unsupervised HIN graph-level embedding algorithms.

- **Financial Card Transactions:** While the field of card transaction research focuses primarily on fraud detection, current graph-based methodologies predominantly utilize simple graphs [RZZ⁺21]. Due to the heterogeneity of the data, this approach does not fully capture the intrinsic complexity of financial networks. This limitation highlights the growing need for HIN-based GNNs for financial transactions. For instance, HINs have been used for the detection of malicious Alipay accounts as a node-level task [LCY⁺18], risky transactions in a B2B network [RZH⁺20], and for bankruptcy prediction in credit risk assessment [ZLWP21]. However, beyond fraud detection, card transaction data is being increasingly leveraged to gain insights into transaction entities. For example, techniques such as text compression have been used to understand the lifestyles of cardholders [DCLOT⁺18]. In addition, graph-based approaches have facilitated the analysis of complex relationships between cardholders and merchants as a node-level task [KRS⁺19], and the refinement of marketing strategies as a node-level task [LWY⁺19]. These diverse applications underscore the versatility of HINs in unraveling complex interactions within heterogeneous financial networks, as comprehensively reviewed in [WZXS22].

3.3 PROBLEM STATEMENT

3.3.1 PRELIMINARIES

The notions of HIN and random metapath walks and discuss the problem addressed here. Table 3.1 shows the mathematical symbols and their definitions used in this chapter.

Symbol	Definition
\mathbb{G}	Dataset of all ego-centric cardholder graphs.
G_k	The k^{th} ego-centric cardholder graph in \mathbb{G} .
$\mathcal{N}_G, \mathcal{R}_G$	Set of nodes and edges in G .
N_G, E_G	Set of node and edge types in G .
τ_N, τ_E	Type mapping functions for nodes and edges.
A_k, D_k, W_k	Adjacency, diagonal, and transition matrices of G_k .
\mathcal{P}, p, π_{mp}	Predefined set of metapaths, a metapath, and its instance.
γ	Teleportation parameter used during random walks.
M	Quantity of negative samples generated during training.
ϕ	Learned graph-level embedding vectors for \mathbb{G} .
\mathbf{h}	Hidden state representation derived from the model.
α_1, α_2	Margins used in the double-triplet loss function.
θ	Set of all parameters in the G-HIN2VEC model.
W_v, W_e, W_G	weight matrices learned for nodes, edges, and graphs.

Table 3.1: Mathematical Symbols and Definitions

Definition 26. *Heterogeneous Information Network, Heterogeneous Information Network*, denoted *HIN*, is defined as a network $G = (\mathcal{N}_G, \mathcal{R}_G, \tau_N, \tau_E, W)$ where G is a weighted directed graph, where \mathcal{N}_G and \mathcal{R}_G denote the set of nodes and edges, $W \in \mathbb{R}^{|\mathcal{N}_G| \times |\mathcal{N}_G|}$ is the weight matrix, and τ_N and τ_E are type mapping functions for nodes and edges, respectively. In HINs, node types are represented by $\tau_N : \mathcal{N}_G \rightarrow N_G$, where N_G is a set of node types, and the relations between nodes are indicated by $\tau_E : \mathcal{R}_G \rightarrow E_G$, where E_G is a set of edge types.

For HINs, metapaths are used to generate random sequences guided by a predefined schema, defined as follows:

Definition 27. *Random Metapath Walk*. Given the schema of G as defined in Def. 26, a metapath $p \in \mathcal{P}$ is defined as the sequence of triplets (n_i, e_i, n_{i+1}) where $n_i \in N_G$ are the source and target node types, and $e \in E_G$ is the edge type, p is denoted as follows:

$$p : \{(n_0, e_0, n_1) \rightarrow (n_1, e_1, n_2) \dots \rightarrow (n_{l-1}, e_{l-1}, n_l)\} \quad (3.1)$$

An instance $\pi_{mp} \sim p(G)$ is a randomized process that begins at v_i ; $\tau_N(v_i) = n_i$, and at each

iteration moves with respect to a specific sampling strategy $Pr(\cdot)$, to the next v_j . After l iterations, the random metapath walk instance is denoted as the sequence $\pi_{mp} = \{ \langle v_i, e_i, v_{i+1} \rangle \sim Pr(v_i|v_{i-1}; p \in \mathcal{P}) \}_{i=0}^l$ following the naive sampling strategy, defined as a uniform random walk constrained by neighboring node types according to the metapath schema, similar to the approaches in [GL16, PARS14]. This ensures that each step of the walk follows the heterogeneous relationships defined by the network schema.

$$Pr(v_i|v_{i-1}; p \in \mathcal{P}) = \begin{cases} \frac{1}{|\mathcal{N}_{n_i}(v_{i-1})|} & e_i \in E \text{ and } \tau_E(e_i) = r_i \\ 0, & \text{otherwise.} \end{cases} \quad (3.2)$$

where $\mathcal{N}_{n_i}(v)$ denotes the neighbors of v of type n_i , and v_i denotes the node i^{th} in the metapath sequence and $e_i = (v_{i-1}, v_i)$. It is important to mention that n_i and r_i denote the node and relation type of the i^{th} element in the metapath schema in p , respectively. A metapath instance π_{mp} is completed after l iterations, ensuring that the start and end node types match ($\tau_N(n_0) = \tau_N(n_l)$), as required by the schema. This symmetry signifies a structured path through the network, following the sampling strategy of Equation 3.2. If the path length l allows, the sequence is considered valid and reflects the relationships defined by the metapath p ; otherwise, it's discarded as an incomplete. The recursive aspect of the walk, $Pr(v_{l+1}|v_l; p \in \mathcal{P}) = Pr(v_0|v_l; p \in \mathcal{P})$, allows the process to be continuous, maintaining the integrity of the network structure. This methodology is common in HG [GL16, PARS14] and adapted for heterogeneous graphs in [DCS17].

3.3.2 PROBLEM DEFINITION

Problem Analysis and Definition for Cardholder Transactions. Instead of using a tripartite large transaction network to model the problem as a node-level representation [KRS⁺19], we propose to build an ego-centric cardholder graph dataset \mathbb{G} to analyze and uncover collective hidden behavior patterns as a graph-level representation problem.

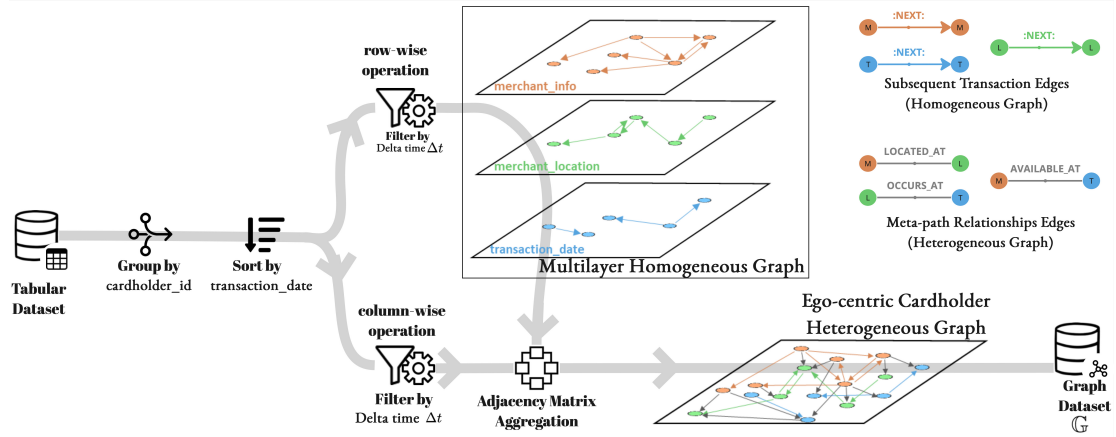


Figure 3.1: Transformation of credit card transactions into an ego-centric graph, layering graphs and metapaths to capture financial interaction patterns.

Definition 28. Ego-centric cardholder graph. Given a set of graphs $\mathbb{G} = \{G_i\}_{i=1}^n$, G_k is the k^{th} heterogeneous ego-centric cardholder graph, including merchant names and code categories (MCC), locations, and discretized time units [DELSoo], as nodes as nodes $\mathcal{N}_{G_k} \subset \mathcal{N}_G$,

$$\mathcal{N}_G = \bigcup_c^{cols} \mathcal{N}_{G_c} \quad (3.3)$$

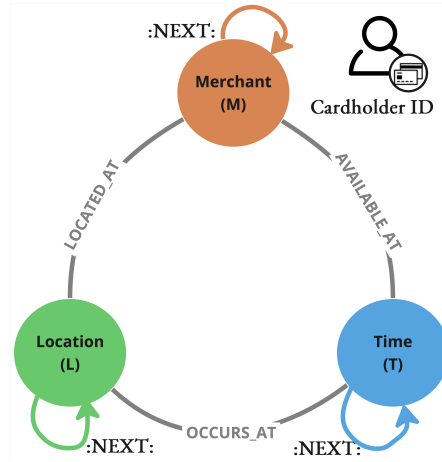


Figure 3.2: Ego-centric cardholder heterogeneous graph schema

The term "cols" refers specifically to the following attributes within the transaction data: `merchant_info`, `merchant_location`, and `transaction_date`. A detailed description of these fields is provided in Section 3.5.1. The cardholder's transaction is the edge $\mathcal{R}_{G_k} = \{(v_i, v_j) \in \mathcal{N}_{G_k} \times \mathcal{N}_{G_k}, w_{i,j}^k \in \mathcal{W}_k : (v_i, v_j, w_{i,j}^k)\}$ and the cardholder's transition matrix $\mathcal{W}_k \in [0, 1]^{V_k \times V_k}$ is defined from the adjacency matrix G_k denoted A_k , where $[A_k]_{ij}$ represents the total consecutive pairs of transactions within a specific time window Δt , set to 24 hours, if the node types are the same in the cardholder's historical transaction records,,

$$[A_k]_{ij} = \underbrace{\sum_{\tau_N(v_i) = \tau_N(v_j)} [1]_{\Delta t}}_{\text{row-wise operation}} + \underbrace{\sum_{\tau_N(v_i) \neq \tau_N(v_j)} 1}_{\text{column-wise operation}} \quad (3.4)$$

The transition matrix \mathcal{W}_k is refined with a 2-hop step with a column stochastic matrix $\mathcal{W}_k = (A_k D_k^{-1})^2$, where D_k is the diagonal matrix with $D_{ij} = \sum_{j=1}^{|\mathcal{N}_{G_k}|} A_{k,(i,j)}$. We amplify the signal in historical data by refining the transition matrix as a weight matrix to enrich the topological and semantic features in G_k . To avoid any confusion, we refer to self-loops in the graph schema as 'next transaction' edges, denoted by :NEXT:, as shown in Figure 3.2.

Problem Definition. After defining the ego-centric cardholder graph dataset \mathbb{G} , we aim to learn the vector representations $\varphi \in \mathbb{R}^{|\mathbb{G}| \times d}$ for every graph $G_i \in \mathbb{G}$, where d is the embedding vector size, $d > 0$.

Problem 1. (Learn from graph substructures) Given a graph $G_k \in \mathbb{G}$, and a set of metapath instances $\pi_{\text{mp}} \sim p$, learn an embedding function $f_G : G_k \rightarrow \mathbf{h}_{G_k} \in \mathbb{R}^d$ that preserves the substructure properties of G_k in the latent graph embeddings space φ .

Problem 2. (Avoid aggregation functions) The aggregation function operates on graph substructures to represent G_k using average, sum, or max pooling layers; this technique ignores the graph topology. Given a graph $G_k \in \mathbb{G}$, learn φ that preserves the proximity between similar graphs in \mathbb{G} and avoid explicit proxy aggregation functions that operate on latent substructure graph embedding.

3.4 OUR APPROACH

Our framework is designed based on an intuitive analogy of graph-level embeddings to define a custom loss function. In this section, we first introduce our motivation and present G-HIN2VEC as an unsupervised learning framework for heterogeneous graphs.

3.4.1 MOTIVATION

The intuition behind our framework is based on a simple analogy with graph kernels for graph-level embeddings. Given two graphs G_i and G_j , for each substructure of an atomic graph a distribution-based measure g is quantified as the similarity between two graphs,

$$D(G_i, G_j) = \sum_{v=1}^U g(\mathbf{h}_{i,v}, \mathbf{h}_{j,v}) \quad (3.5)$$

where $U = \mathcal{N}(G_i) \cap \mathcal{N}(G_j)$ is the set of common substructures in both graphs. In this case nodes where $\mathcal{N}(G_i)$ represent nodes of G_i and $\mathbf{h}_{i,v}$ could indicates the color of the node v in G_i for WL-OA [SSVL⁺11], or a random graphlet starting from v for graphlet kernel [BK05] or a subgraph where the ego node is v for SP-kernel [SVP⁺09], and D is considered as a graph representation kernel. In GNNs, g is applied to atomic graph substructures and D is considered as a graph representation,

$$D(G_i, \emptyset) = \sum_{v=1}^U g(\mathbf{h}(i, v), \emptyset) = \sum_{v=1}^{N_{G_i}} w_v \mathbf{h}(i, v) \quad (3.6)$$

Here, g could be a static pooling operation such as average operation (where $w_v = \frac{1}{|N_{G_i}|}$) or a learned aggregation layer [YYM⁺18]. Regardless of the local properties encoder function Q could be added and applied to \mathbb{G} to represent G_i by an n -dimensional vector representing

the distances from all graphs.

$$Q(G_i) = \bigcup_{G_j}^{\mathbb{G}} \{D(G_i, G_j)\} \quad (3.7)$$

In Equation 3.5 and 3.6, g is seen as a distance function that quantifies the dispersion within and between graphs for different local D and global Q representations. Our goal is to learn the graph representation by quantifying this dispersion without the need for explicit aggregation of the graph substructures. Simultaneously, our goal is to avoid using the true distance matrix (Equation 3.7) for global properties.

3.4.2 G-HIN₂VEC EMBEDDING LEARNING

The key novelty of G-HIN₂VEC is to determine if a metapath instance π_{mp} from a given semantic context $p \in \mathcal{P}$ is a subset of the heterogeneous graph G_i , “Does $\pi_{\text{mp}} \subseteq G_i$?”, thereby learning graph proximity through intra-graph substructures.

Given \mathbb{G} and $p^+ \in \mathcal{P}$, the objective is to maximize the probability,

$$\arg \max_{\theta} \prod_{G_i \in \mathbb{G}} \prod_{\pi_{\text{mp}} \in p^+(G_i)} Pr(\pi_{\text{mp}}^{(\cdot)} | G_i; \theta) \quad (3.8)$$

where $\pi_{\text{mp}}^{(\cdot)} = \{ \langle v_{i,k}, e_{i,k}, v_{i,k+1} \rangle \}_{k=0}^l$ is a sequence of triplets with respect to p^+ , which can be defined as a semantic substructures of G_i , and $Pr(\pi_{\text{mp}}^{(\cdot)} | G_i; \theta)$ represents the conditional probability of having $\pi_{\text{mp}}^{(\cdot)}$ given G_i .

To learn a graph representation $\varphi \in \mathbb{R}^{|\mathbb{G}| \times d}$, we made a simple extension to the heterogeneous skip-gram model presented in [DCS17], to incorporate the metapath schema to learn

from graph substructure by maximizing logarithmic probability,

$$\arg \max_{\theta_1, \theta_2} \sum_{G_i \in \mathbb{G}} \sum_{\pi_{\text{mp}} \in p^+(G_i)} \sum_{t=1}^l \log(Pr(\pi_{\text{mp}}^{(t+)} | G_i; \theta_1) Pr(\pi_{\text{mp}}^{(\neq t+)} | G_i; \theta_2)) \quad (3.9)$$

Here, it is assumed that the positive metapath context instance $\pi_{\text{mp}}^{(\neq t+)} = \{ \langle v_{i,k}, e_{i,k}, v_{i,k+1} \rangle \}_{k=0, k \neq t}^l$ and the target instance $\pi_{\text{mp}}^{(t+)} = \{ \langle v_{i,t}, e_{i,t}, v_{i,t+1} \rangle \}$ are independent for a given G_i , where $\pi_{\text{mp}}^{(t+)}$ represents the t^{th} triplet in the metapath sequence and $\pi_{\text{mp}}^{(\neq t+)}$ represents the entire sequence excluding the t^{th} triplet.

In general, a heterogeneous graph requires more than one metapath to capture rich semantics; for this, G-HIN2VEC expands Equation 3.9 to cover the entire set of predefined metapaths \mathcal{P} ,

$$\arg \max_{\theta_1, \theta_2} \sum_{G_i \in \mathbb{G}} \sum_{p^+ \in \mathcal{P}} \sum_{\pi_{\text{mp}} \in p^+(G_i)} \sum_{t=1}^l \log Pr(\pi_{\text{mp}}^{(t+)} | G_i; \theta_1) + \log Pr(\pi_{\text{mp}}^{(\neq t+)} | G_i; \theta_2) \quad (3.10)$$

where $Pr(\pi_{\text{mp}}^{(\cdot)} | G_i)$ is defined as

$$\frac{\exp(\mathbf{h}_{\pi_{\text{mp}}^{(\cdot)}}^+ \cdot \mathbf{h}_{G_i})}{\sum_{G_j \in \mathbb{G}} \sum_{\pi_{\text{mp}_j} \in p^+(G_j)} \exp(\mathbf{h}_{\pi_{\text{mp}_j}^{(\cdot)}}^+ \cdot \mathbf{h}_{G_j})} \quad (3.11)$$

In order to train our G-HIN2VEC model more efficiently and avoid the exhaustive computation of considering every substructure within the entire graph dataset for each metapath in \mathcal{P} and target element in $\pi_{\text{mp}}^{(\cdot)}$, we adopt a negative sampling technique. Specifically, instead of using negative samples from different graphs in the dataset, we generate negative samples directly within the graph of interest using Algorithm 1. This allows us to create more contextually relevant and computationally efficient negative instances. For every positive instance, we generate 5 negative instances, setting the M negative sample instances to 5 for each positive instance, thus maintaining a 1:5 ratio. This ratio is a tunable hyperparameter, and while

our current infrastructure supports a 1:5 ratio, this can be adjusted according to the scale and capacity of the training setup. The sampling distribution for each metapath in \mathcal{P} is carefully specified, drawing inspiration from [SHH⁺18]. In the same direction, Equation 3.10 can be expressed as a distance-based objective function following Equation 3.5 and 3.6 to customize the loss function introduced in [FLL17][NCV⁺17]. Therefore, we have the following objective:

$$\begin{aligned} \mathcal{L} = & \sum_{\substack{\pi_{\text{mp}}^+ \sim p^+(G_i) \\ \pi_{\text{mp}}^- \sim p^-(G_i) \\ (p^+, p^-) \sim \mathcal{P} \\ p^+ \neq p^-}} \underbrace{\left[g_{nn}(\mathbf{h}_{\pi_{\text{mp}}}^+, \mathbf{h}_{G_i}^+)^2 - g_{nn}(\mathbf{h}_{\pi_{\text{mp}}}^-, \mathbf{h}_{G_i}^-)^2 + \alpha_1 \right]}_{\substack{\text{Node} \\ \text{Triplet} : \text{Triplet node loss}}} + \\ & + \underbrace{\left[f_{nn}(\mathbf{h}_{\pi_{\text{mp}}}^+, \mathbf{h}_{G_i}^+)^2 - f_{nn}(\mathbf{h}_{\pi_{\text{mp}}}^-, \mathbf{h}_{G_i}^-)^2 + \alpha_2 \right]}_{\substack{\text{Graph} \\ \text{Triplet} : \text{Triplet graph loss}}} \end{aligned} \quad (3.12)$$

where $[\cdot]_+ = \max(\cdot, 0)$, and $p^- \sim_{\text{unif.}} \mathcal{P}$ is the negative metapath and π_{mp}^- is the negative metapath instance defined as a random sequence of triplets with respect to p^- . Note that π_{mp}^- does not necessarily exist in \mathbb{G} following Algorithm 1.

Hidden substructure representations are the output of a shared weight network for each triplet element, nodes and edge, $\mathbf{h}_{v_k} = \sigma(W_v \vec{x}_{v_k})$ and $\mathbf{h}_{e_k} = \sigma(W_e \vec{x}_{e_k})$, respectively, with $\vec{x}_v \in \mathbb{R}^{|\mathcal{N}_G|}$ and $\vec{x}_e \in \mathbb{R}^{|\mathcal{E}|}$ are the one-hot indicator vectors, and $W_v \in \mathbb{R}^{|\mathcal{N}_G| \times d}$ and $W_e \in \mathbb{R}^{|\mathcal{E}| \times d}$ are weight matrices. The same applies for the hidden representation of the graph $\mathbf{h}_{G_i} = \sigma(W_G \vec{x}_{G_i})$, with $\vec{x}_{G_i} \in \mathbb{R}^{\mathbb{G}}$ and $W_G \in \mathbb{R}^{\mathbb{G} \times d}$ are the embedding and weight matrices, $\sigma(\cdot)$ is the sigmoid function. Then a metapath-type-aware concatenation mechanism is used to stabilize the learning process with respect to the heterogeneity of metapaths as a triplet concatenation operation $\mathbf{h}_{\pi_{\text{mp}}^{(\cdot)}} = \text{Concat}(\pi_{\text{mp}}^{(\cdot)})$, defined as

$$\mathbf{h}_{\pi_{\text{mp}}^{(\cdot)}} = W_p \left(\prod_{k=1}^l \left(\frac{1-\beta}{2} \mathbf{h}_{v_k} \right) \odot (\beta \mathbf{h}_{e_k}) \odot \left(\frac{1-\beta}{2} \mathbf{h}_{v_{k+1}} \right) \right) \quad (3.13)$$

where $\mathbf{h}_{\pi_{\text{mp}}^{(\cdot)}} \in \mathbb{R}^c$ is the hidden representation of $\pi_{\text{mp}}^{(\cdot)}$ and $W_p \in \mathbb{R}^{d \times c}$ is a linear transformation of a nonlinear function to project the hidden representations of the substructure to a

specific output dimension, and $\beta \in [0, 1]$ is a signal amplifier for triplet relations, set to 0.4. Same for graph hidden representation where $W_p \mathbf{h}_{G_i} \in \mathbb{R}^c$.

Equation 3.13 is a metapath-aware operation used for similarity measurement, without an activation function.

In summary, given the projected feature vector for G_i and $\pi_{mp}^{(\cdot)}$ positive and negative contexts, we learn two distinct functions to measure the similarity between the hidden representations of the graph and nodes, $\langle \mathbf{h}_{\pi_{mp}^{t+}}^+, \mathbf{h}_{G_i}^+ \rangle$ and $\langle \mathbf{h}_{\pi_{mp}^{t-}}^-, \mathbf{h}_{G_i}^- \rangle$, respectively, with $g_{nn}(\cdot)$, and between the hidden representations of the graph and the metapath instance, $\langle \mathbf{h}_{\pi_{mp}^{t+}}^+, \mathbf{h}_{G_i}^+ \rangle$ and $\langle \mathbf{h}_{\pi_{mp}^{t-}}^-, \mathbf{h}_{G_i}^- \rangle$, respectively, with $f_{nn}(\cdot)$. Following [WZL⁺16], both $g_{nn}(\cdot)$ and $f_{nn}(\cdot)$ are fully connected layers with a one-dimensional output, instead of using static functions such as Euclidean distance or cosine similarity.

As $g_{nn}(\cdot)$ and $f_{nn}(\cdot)$ represents a learned metric where the largest the value more the hidden representations are dissimilar, therefore, the dissimilarity and probability defined in Equation 3.10 must be correlated positively with both functions. Similarly to [WZL⁺16], we define both functions with a softmax activation layer to normalize the output in $[0, 1]^2$ and keep only one dimension as a dissimilarity value formulated as follows:

$$\begin{cases} g_{nn}(\mathbf{h}_{\pi_{mp}^{(\cdot)}}^+, \mathbf{h}_{G_i}^+) = \text{softmax}(W_g^T \cdot (\mathbf{h}_{\pi_{mp}^{(\cdot)}}^+, \mathbf{h}_{G_i}^+) + B_g)_0 \\ f_{nn}(\mathbf{h}_{\pi_{mp}^{(\cdot)}}^+, \mathbf{h}_{G_i}^+) = \text{softmax}(W_f^T \cdot (\mathbf{h}_{\pi_{mp}^{(\cdot)}}^+, \mathbf{h}_{G_i}^+) + B_f)_0 \end{cases} \quad (3.14)$$

where $W_g \in \mathbb{R}^{2c \times 2}$, $W_f \in \mathbb{R}^{2c \times 2}$, $B_g \in \mathbb{R}^{2c \times 1}$, and $B_f \in \mathbb{R}^{2c \times 1}$, are the weight matrices for the learned metric function g_{nn} and f_{nn} and their bias terms, respectively.

Algorithm 1: MetapathSeqNoising Function.**Input** : $\pi_{\text{mp}}^+, p^-, l, \lambda$ **Output** : π_{mp}^- negative metapath instance. $l' = \lceil \lambda l \rceil$; // Number of noise edges ; $(v_t^-, v_{t+1}^-) \leftarrow \text{Shuffle}(\{v \in V \mid \tau_N(v) \in p^-\}, l')$; $e_t^- \leftarrow \text{Shuffle}(\{e \in E \mid \tau_E(v_t^-, v_{t+1}^-) \in p^-\}, l')$;// Noise injection in π_{mp}^+ as a concatenation process; $l_1 = \lceil \frac{l-l'}{2} \rceil$ and $l_2 = l - l_1$; $\pi_{\text{mp}}^- \leftarrow \{\pi_{i,k}^+\}_{k=1}^{l_1} \oplus \{< v_t^-, e_t^-, v_{t+1}^- >\}_{t=1}^{l'} \oplus \{\pi_{i,k}^+\}_{k=l_2+1}^l$;**Return** : π_{mp}^-

In our loss definition in Equation 3.12, we employ two margin threshold terms, α_1 and $\alpha_2 \in [0, 1]$, to modulate the similarity and dissimilarity within and between graph representations. Specifically, α_1 , set to 1, regulates intra-graph similarity, ensuring that positive samples of nodes and substructures within the same graph are embedded closely together in the embedding space. On the other hand, α_2 , set to 0.6, governs inter-graph dissimilarity, ensuring that negative samples of nodes and substructures are sufficiently separated in the embedding space. In the unsupervised learning context, where graph labels are unavailable, we refine the weights of G-HIN2VEC by minimizing a double-triplet loss function. This process is part of our training data preparation, which aims to generate embeddings for the graph dataset, nodes, and relations. These embeddings are denoted as $\varphi \in \mathbb{R}^{|\mathbb{G}| \times d}$ for the graph-level, $X_v \in \mathbb{R}^{|\mathcal{N}_G| \times d}$ for nodes, and $X_r \in \mathbb{R}^{|\mathcal{R}_G| \times d}$ for relations. Our focus is to generate a representation at the graph-level φ .

3.4.3 TRAINING DATA PREPARATION

Algorithm 2: G-HIN₂VEC training algorithm.

Input : The heterogeneous graph dataset \mathbb{G} ,
 metapaths $\mathcal{P} = \{p_1, \dots, p_k\}; k > 0$,
 global learned weight W_j , batch size B
 walk length l , negative samples M , noise level λ

Output : The global weight W_{i+1} matrix.

$G = \text{Shuffle}(\mathbb{G}, B)$; // Shuffle and Random B samples.

for $i = 1, \dots, B$ **do**

$p^+ = \text{Shuffle}(p, 1)$;

$p^- = \text{Shuffle}(\{p \in \mathcal{P} | p \neq p^+\}, M)$;

$\pi_{\text{mp}}^+ := \text{MetapathSeq}(G_i, p^+, l)$;

for $m = 1, \dots, M$ **do**

// M random negative samples, set to ζ .

$\pi_{\text{mp}}^- = \text{MetapathSeqNoising}(\pi_{\text{mp}}^+, p^-, l, \lambda)$;

$\mathbf{h}_{G_i} = \sigma(W_{G_i} \vec{x}_{G_i})$;

for $k = 1, \dots, l$ **do**

// For each Triplet in π_{mp}^+ and π_{mp}^- ;

$\langle v_{i,k}^+, e_{i,k}^+, v_{i,k+1}^+ \rangle \leftarrow \pi_{\text{mp}}^+$;

$\langle v_{i,k}^-, e_{i,k}^-, v_{i,k+1}^- \rangle \leftarrow \pi_{\text{mp}}^-$;

// Relation hidden representation transformations;

$\mathbf{h}_{e_{i,k}}^+ = \sigma(W_{e_{i,k}} \vec{e}_{i,k}^+)$;

$\mathbf{h}_{e_{i,k}}^- = \sigma(W_{e_{i,k}} \vec{e}_{i,k}^-)$;

// Node hidden representation transformations;

$\mathbf{h}_{v_{i,k}}^+, \mathbf{h}_{v_{i,k+1}}^+ = \sigma(W_{v_{i,k}} \vec{v}_{i,k}^+), \sigma(W_{v_{i,k+1}} \vec{v}_{i,k+1}^+)$;

$\mathbf{h}_{v_{i,k}}^-, \mathbf{h}_{v_{i,k+1}}^- = \sigma(W_{v_{i,k}} \vec{v}_{i,k}^-), \sigma(W_{v_{i,k+1}} \vec{v}_{i,k+1}^-)$;

end

$t = \text{Shuffle}(\text{interval}[1, l], 1)$ // Sample target context ;

// Representations alignment and concatenation;

$\mathbf{h}_{G_i}^+ = W_{p^+} \mathbf{h}_{G_i}$; $\mathbf{h}_{G_i}^- = W_{p^-} \mathbf{h}_{G_i}$;

$\mathbf{h}_{\pi_{\text{mp}}^+}^+ = \text{Concat}(\mathbf{h}_{\pi_{\text{mp}}^+}^+)$; $\mathbf{h}_{\pi_{\text{mp}}^-}^- = \text{Concat}(\mathbf{h}_{\pi_{\text{mp}}^-}^-)$;

$\mathbf{h}_{\pi_{\text{mp}}^+}^{\neq t+} = \text{Concat}(\mathbf{h}_{\pi_{\text{mp}}^+}^{\neq t+})$; $\mathbf{h}_{\pi_{\text{mp}}^-}^{\neq t-} = \text{Concat}(\mathbf{h}_{\pi_{\text{mp}}^-}^{\neq t-})$;

// Output layer as learned dissimilarity metric;

$d_{\text{node}}^+, d_{\text{node}}^- = g_{nn}(\mathbf{h}_{\pi_{\text{mp}}^+}^+, \mathbf{h}_{G_i}^+)^2, g_{nn}(\mathbf{h}_{\pi_{\text{mp}}^-}^-, \mathbf{h}_{G_i}^-)^2$;

$d_{\text{mp}}^+, d_{\text{mp}}^- = f_{nn}(\mathbf{h}_{\pi_{\text{mp}}^+}^{\neq t+}, \mathbf{h}_{G_i}^+)^2, f_{nn}(\mathbf{h}_{\pi_{\text{mp}}^-}^{\neq t-}, \mathbf{h}_{G_i}^-)^2$;

// Cost function Equation 3.12;

$\text{Cost}_i = [d_{\text{node}}^+ - d_{\text{node}}^- + \alpha_1]_+ + [d_{\text{mp}}^+ - d_{\text{mp}}^- + \alpha_2]_+$;

end

end

$W_{i+1} := \text{miniBatchSGD}(W_i, \text{Cost}_i)$ // Weight updates;

As introduced previously, G-HIN2VEC uses the negative sampling technique to generate training data. Therefore, data preparation is a crucial phase in our approach. As detailed in Algorithm 2 and illustrated in Figure 3.3, we train our model in mini-batches, sampled from \mathbb{G} , and then generate the positive context π_{mp}^+ from G_i with a predefined metapath $p^+ \sim \mathcal{P}_{\text{unif.}}$ following our custom sampling strategy.

$$Pr(v^i | v^{i-1}; p^+) = \begin{cases} \beta \frac{1}{|\mathcal{N}_{n_i}(v^{i-1})|} & e_i \in \mathcal{R}_G \text{ and } \tau_E(e_i) = r_i \\ (1 - \beta) \frac{1}{|\mathcal{N}_{\neq n_i}(v^{i-1})|} & e_i \in \mathcal{R}_G \text{ and } \tau_E(e_i) \neq r_i \\ 0 & e_i \notin \mathcal{R}_G; \text{ where } e_i = (v^i, v^{i-1}) \end{cases} \quad (3.15)$$

where β is the teleportation term to escape nodes with high centrality and $\mathcal{N}_{\neq n_i}(v)$ denotes the neighbor v^i of different types of nodes $n_i \in p^+$, denoted by *MetapathSeq* in Algorithm 1.

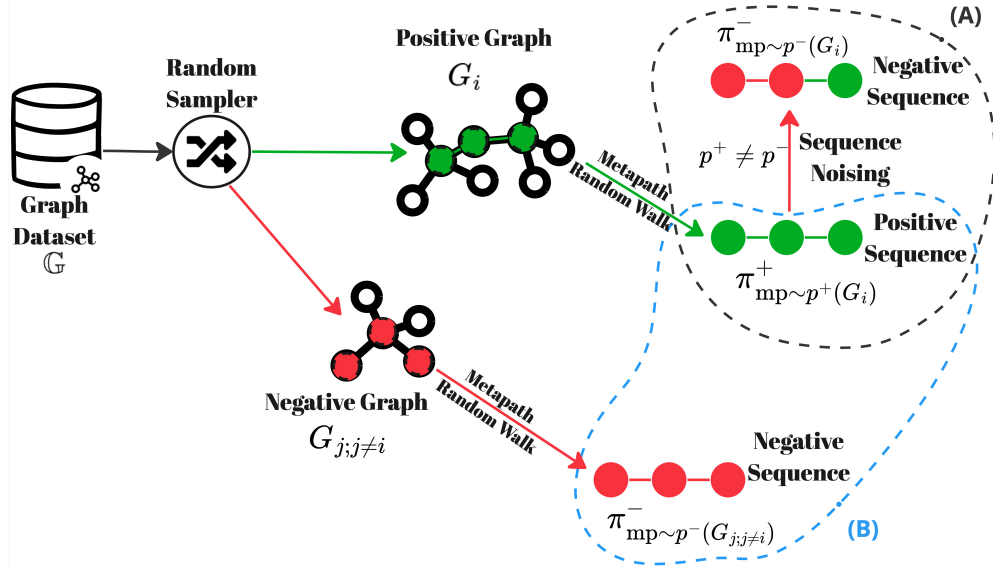


Figure 3.3: G-HIN2VEC training data generation process

As in Figure 3.3, our approach introduces noise to the original positive metapath instances π_{mp}^+ , creating negative contexts π_{mp}^- within the same graph rather than from different graphs. This is achieved by perturbing π_{mp}^+ with alternative metapaths p^- uniformly sampled from

\mathcal{P} . The noise level $\lambda \in [0, 1]$ is set to 0.3; a process we denote as *MetapathSeqNoising*. After generating both positive and negative instances, we apply a series of linear transformations to derive the final triplet hidden representations to estimate the dissimilarity metrics.

Depending on the quantified metrics in our unsupervised framework, we optimize the global model weights by minimizing the double-triplet loss via back-propagation and gradient descent to learn meaningful graph-level embeddings for the heterogeneous graph dataset.

3.5 DESIGN OF EXPERIMENTS

In this section, we evaluate the proposed model in a real-world credit card transaction dataset from a European bank. Our empirical analysis focuses on qualitative and quantitative analyses. It is important to mention that the dataset used in the current research work is fully aligned with the General Data Protection Regulation (GDPR) in the European Union to protect personal data as defined in Article 4.

3.5.1 DATASETS AND TASKS

We conducted our experiments on a dataset comprising records from 100K cardholders and transactions from 2019 to 2021. The dataset's schema includes the transaction table: `cardholder_id` (as unique INT), `merchant_info` (as VARCHAR), representing merchant code category and merchant name such as '5661-Shoe Stores', `merchant_location` (as VARCHAR), detailing the location like 'Paris-France', and `transaction_date` (as DATE), indicating when the transaction occurred, for instance, '2019-06-21'. This raw data was transformed into an ego-centric cardholder graph dataset, as elaborated in Section 3.3. Furthermore, we used the cardholder table: `cardholder_id` (as unique INT), `Gender` (as VARCHAR), `Age` (as FLOAT) and `Income` (as FLOAT). All types are according to SQL data types.

The cardholder socio-demographic attributes used for supervised machine learning experiments include:

- **Gender**, $Y_{Gender}^i \in \{0, 1\}, \forall i \in \{1, \dots, n\}$ as a binary attribute indicating the i^{th} cardholder's gender, where 0 represents one gender and 1 represents another.
- **Income**, $Y_{Income}^i \in [0, 1], \forall i \in \{1, \dots, n\}$ as a normalized attribute representing the i^{th} cardholder's income.
- **Age**, $Y_{Age}^i \in [0, 1] \forall i \in \{1, \dots, n\}$ as a normalized attribute representing the i^{th} cardholder's age.

For quantitative analysis, we performed a binary classification task on Y_{Gender} and a regression task on Y_{Age} and Y_{Income} to benchmark different hidden representation models on predictive tasks for the ego-centric graph of cardholders. On the other hand, we also performed a qualitative analysis based on cardholder representations as a cluster analysis task to illustrate and interpret hidden communities.

3.5.2 BASELINES AND EXPERIMENTAL SETTINGS

In this section, we outline the baseline methods used for comparison with our G-HIN2VEC approach, focusing on homogeneous and heterogeneous graph embeddings, as well as other graph-level representation techniques. These baselines are detailed in Table 3.2 and grouped into three primary categories. Our G-HIN2VEC method is implemented using the StellarGraph framework, a Python library for machine learning on graphs. All graph embeddings for the cardholder were learned using G-HIN2VEC and the baseline models. For G-HIN2VEC specifically, the experimental settings include a dropout rate of 0.5, and the use of the SGD optimizer. We adjust the margin thresholds α_1 and α_2 in our loss function, in Equation 3.12, to 1 and 0.5, respectively, with a learning rate of 0.005. Additionally, weight decay is accompanied by an L2 penalty set at 0.001 over 50 epochs.

Category	Category Description	Method	Method Description
Vectorization-based	Uses node and graph features to represent graphs, focusing on node frequency, graph statistics, and structural properties such as centrality measures, graph density, and spectral properties.	N-grams ($N = 1, 2$)	Captures local graph features through individual and pairs of nodes frequency, emphasizing simpler patterns.
		Bag-of-Features (BoF)	Incorporates comprehensive graph features, including statistics on node degree, clustering coefficients, and triangles and squares sub-graph counts.
Kernel-based	Employing graph kernels, such as outlined in Equation 3.5 to map graphs into a vector space. This transformation facilitates the computation of a distance matrix for comparing graphs, as illustrated in Equation 3.7.	Graphlet kernel (GK) [BK05]	Represents graphs based on the frequency of small subgraphs (graphlets) throughout the graph dataset, ideal for capturing local topology.
		Shortest path kernel (SPK) [SVP ⁺ 09]	Encodes each graph by the shortest path lengths between nodes within the graph dataset, reflecting global connectivity.
		Weisfeiler-Lehman framework (WL-G) [SSVL ⁺ 11]	Refines graph representation through iterative relabeling based on neighborhood structures to capture structural changes over multiple scales, reflecting both local and global structural differences throughout the graph dataset.
		Deep GK, Deep SP, Deep WL [YV15a]	Apply deep learning techniques with traditional graph kernels to generate graph-level embeddings. All variants capture different complex structural graph features in the graph dataset.
GNN-based	Uses deep learning architectures to generate node-level and graph-level embeddings, for node-level architectures an aggregation function is used to produce graph-level embeddings from node-level from node-level and edge-level embeddings, as in Equation 3.6.	Metapath2vec [DCS17]	Uses guided random walks by metapaths in heterogeneous graphs to generate node embeddings. For graph-level features, the graph node embeddings are then aggregated using mean to produce graph-level embeddings.
		HIN2VEC [FLL17]	Generate node and relation embeddings in heterogeneous networks by considering multiple types of nodes and relationships. For graph-level features, graph nodes and relationships are aggregated by average to form graph-level embeddings.
		Graph2Vec [NCV ⁺ 17]	Treat the entire graph as an individual document in a corpus, employing a document embedding approach to learn graph embeddings. This method captures global graph properties and structural features.
		DiffPool [YYM ⁺ 18]	Implement a differentiable graph pooling module that hierarchically aggregates nodes embeddings into clusters, forming a new, coarser graph at each layer, until final cluster as the graph-level representations.

Table 3.2: Detailed overview of graph-based methods categorized by their approach and core features, including graph-level and heterogeneity, with a focus on graph representations.

In this section, we utilized traditional graph features including node degree distribution statistics (mean, median, standard deviation, skewness, kurtosis), graph density [Wes01], global and average local clustering coefficients [WS98], number and size of connected components [Tar72], graph diameter and average path length [Flo62], centrality measures (degree, be-

tweenness, eigenvector) [Fre79][Bon87], assortativity [New03], spectral properties (Laplacian eigenvalues, spectral gap) [Chu97], subgraph counts (triangles, squares) [Har73], and graphlet frequencies [Prz07].

After the pretraining phase on the graph dataset of cardholders, we benchmarked the performance of the generated embeddings on a set of downstream tasks. Linear and logistic regression were implemented on the graph embedding vector for regression and classification tasks, respectively. The dataset was divided into training, validation, and test sets based on the cardholder ID to ensure that all transactions related to a single cardholder fall into the same group. The split ratio was 2:1:1 for training, validation, and test sets, respectively. The performance results reported on Table 3.4 are the average of 10-fold cross-validation test sets, with the statistical significance of each metric validated by calculating the p-value. Performance for graph-level representation in the classification task was evaluated using averaged accuracy, AUC, and F1 metrics, and for regression tasks using R-squared (R^2) and mean absolute error (MAE) metrics.



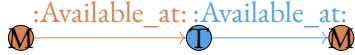




BQ 1:	What spending habit does the cardholder have?
Metapath 1	
BQ 2:	Where does the cardholder use the credit card?
Metapath 2	
BQ 3:	When does the cardholder use the credit card?
Metapath 3	
BQ 4:	When and where is the credit card used?
Metapath 4	

Table 3.3: Set of business questions as metapaths by financial experts, where  Merchant,  Location, and  Time stand for node types.

3.6 EXPERIMENTS AND RESULTS

3.6.1 EMPIRICAL VALIDATION

We performed empirical validation on the graph dataset following the experimental settings described earlier. For non-graph-level representation models such as HIN2Vec, we added a simple averaging layer on top of each node-level representation to generate graph representations. In heterogeneous guided random walk-based GNNs, the definition of the metapath is required.

For this purpose, experts provided a set of metapaths to answer specific business questions (BQ), as shown in Table 3.3. Furthermore, metapath-free baselines ignore the predefined metapaths for both node- and graph-level representation methods, and this will not be considered in both experimental analyses due to the original implementations.

3.6.2 QUANTITATIVE ANALYSIS

Category	Method	Gender		Income		Age	
		Acc.	AUC	R ₂	MAE	R ₂	MAE
Vectorization based	N-gram (1 & 2-grams)	0.6133	0.601	0.1067	0.1531	0.1045	12.7074
	Bag of Features (BoF)	0.7295	0.722	0.1751*	0.1409*	0.2994	10.9955
Kernel based	Graphlet kernel [BK05] (GK)	0.5825	0.5647	0.0449	0.1618	0.0846	12.5174
	Shortest path kernel [SVP ⁺ 09] (SPK)	0.5477	0.5002	0.0698	0.1597	0.0873	12.9465
	Weisfeiler-Lehman [SSVL ⁺ 11] (WL)	0.7001	0.6993	0.2030	0.1410	0.4301	9.8510
Deep Kernel based [YV15a]	Deep GK	0.5399	0.5185	0.0472	0.1469	0.0842	11.4422
	Deep SP	0.5566	0.5137	0.0798	0.1629	0.0922	13.1960*
	Deep WL	0.7032*	0.7292**	0.2255	0.1381*	0.4717	9.5978**
GNN based	Metapath2vec [†] [DCS17]	0.6102	0.5586	0.1105**	0.1489	0.1349	12.3272
	Graph2Vec ^{†§} [NCV ⁺ 17]	0.7220	0.7331	0.2394*	0.1354	0.5384	9.4021
	HIN2VEC [†] [FLL17]	0.7809	0.7906	0.2318	0.1399*	0.6385*	6.6019
	DiffPool ^{†§} [YYM ⁺ 18]	0.8047*	0.8194	0.3088**	0.1212**	0.7108*	5.9277*
	G-HIN2VEC ^{†§}	0.8244*	0.8311*	0.3310*	0.1137**	0.6843**	6.3157*

Table 3.4: Performance of various graph-based methods in predicting cardholder attributes. Superscripts denote model compatibility: [‡] for heterogeneous graphs, [†] for homogeneous graphs, ^{||} for node-level, and [§] for graph-level downstream tasks. Statistical significance is denoted as: ** $p < 0.01$, * $p < 0.05$.

REGRESSION TASKS.

In regression tasks targeting age and income prediction, our evaluation reveals that vectorization and graph kernel-based methods struggle to effectively model the complex relationships inherent in heterogeneous graph data, resulting in inferior performance. In contrast, GNN-based models show enhanced outcomes due to their advanced architectures, which are better suited to capture the complex interactions within these graphs. Notably, our G-HIN2VEC model demonstrates significant improvements over the strong baseline, DiffPool, with a 7.18% increase in R-squared and a 6.19% decrease in mean absolute error for income prediction, despite its non-Gaussian distribution. For age prediction, which follows a Gaussian distribution, G-HIN2VEC matches the performance of DiffPool. These results affirm the robustness of G-HIN2VEC in generating effective graph-level embeddings for regression tasks, highlighting its potential for broader application in complex graph-based analysis.

CLASSIFICATION TASK.

The gender classification performance of various methods is comprehensively detailed in Table 3.4. Among the benchmarked methods, vectorization methods are observed to outperform graph kernel-based methods, showcasing their better suitability for classification tasks involving categorical data. In contrast, among the baselines, DiffPool stands out as the most effective model. It highlights the benefits of utilizing sophisticated aggregation techniques rather than averaging at the top of the node-level, thus enhancing the quality of graph-level embeddings. In the specific case of our G-HIN2VEC model, significant improvements in gender classification are evident, with increases of 2.44% in accuracy, 1.85% in F1 score, and 1.43% in AUC compared to DiffPool. These statistically significant results demonstrate G-HIN2VEC's superior capability in leveraging complex graph-level embeddings for classification tasks.

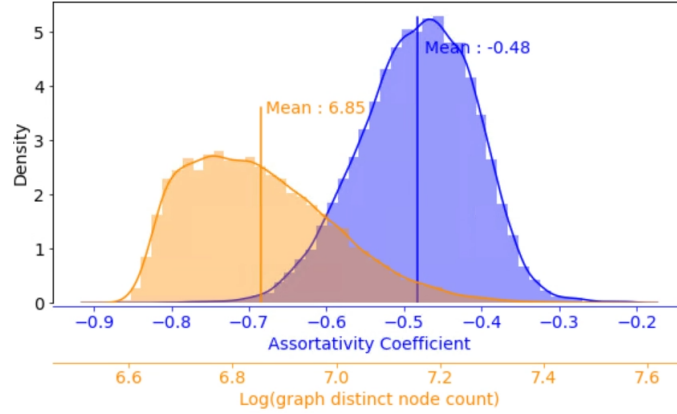


Figure 3.4: Probability distribution of log graph sizes and assortativity coefficient.

Additionally, the analysis of the graph structures within our dataset, particularly their size distribution and negative assortativity coefficients as shown in Figure 3.4, supports these outcomes. This structural characteristic suggests that G-HIN₂VEC not only effectively handles disassortative graphs but also benefits from adjustments such as the incorporation of a teleportation term, β , in our sampling strategy (see Equation 3.15), which enhances model robustness and confirms the quantitative findings of improved performance. Furthermore, a primary limitation of GNN models, as highlighted in [SBN⁺21], involves their sensitivity to assortativity during node-level tasks, which can extend to graph-level tasks, potentially impacting performance; our model’s design addresses and mitigates this limitation effectively.

3.6.3 QUALITATIVE ANALYSIS

In this section, we investigated the cluster assignment of cardholder behavior through the graph-level embeddings. Figure 3.5 presents a t-SNE visualization of cardholder embeddings, where each color signifies a distinct cluster, corresponding to a unique behavioral pattern in credit card usage. Our assumption is that cardholders who are semantically similar in using credit cards tend to be embedded in close proximity, as shown in [NCV⁺17] and [DCS17] for graph- and node-level representations, respectively.

The clustering of embeddings into 16 distinct groups, as determined by the elbow method using k-means, reveals a granular view of cardholder behaviors, with each cluster capturing a particular lifestyle preference that aligns with previous findings in credit card usage research [DCLOT⁺18]. In post-clustering analysis, we engaged domain experts to identify cluster identities, extracting and examining metapath instances that were sampled during training. This analysis aimed to quantify the occurrence of directed sequences within each cluster, thus providing a narrative for the lifestyle categories annotated in Figure 3.5. Thus, two cardholders are semantically similar if their credit card usage behavior is similar with respect to a set of predefined metapaths in Table 3.3.

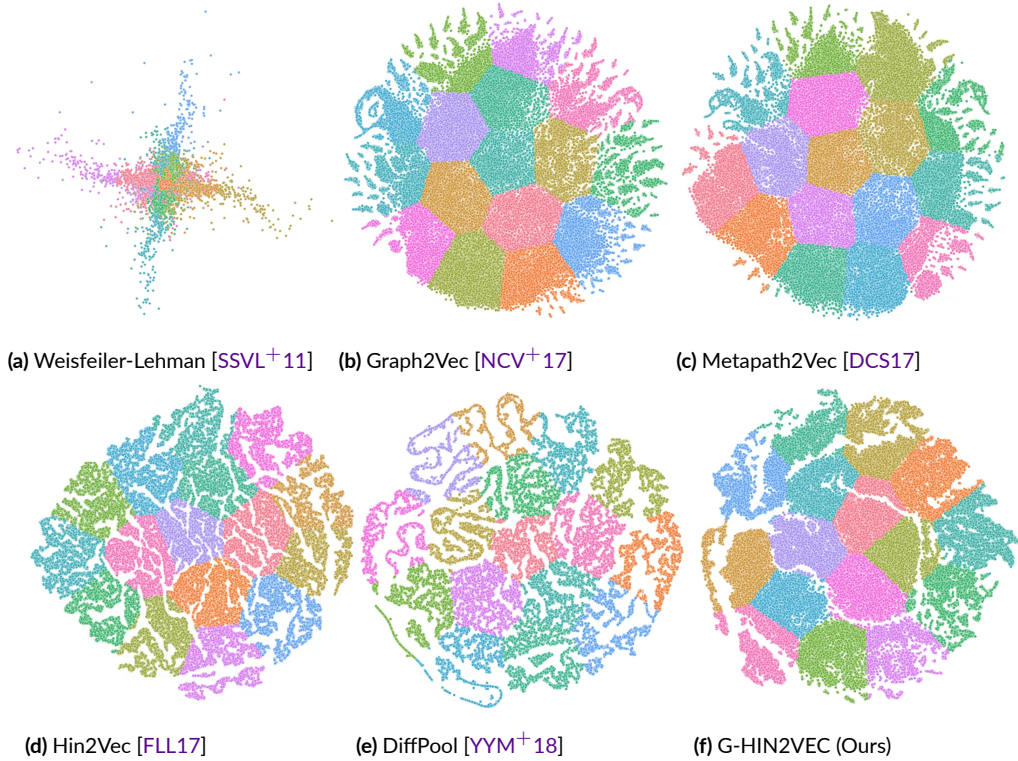


Figure 3.5: t-SNE visualization of graph embeddings, with 16 clusters.

The t-SNE visualizations highlight how well different graph-level embedding methods group similar cardholder behaviors.

The Weisfeiler-Lehman method results in overlapping clusters, suggesting that it may not be as effective in identifying the finer details of the graph-level features. Graph2Vec and Metapath2Vec improve upon this with more defined clusters, reflecting their stronger ability to map out both the individual elements and graph-level structures. HIN2VEC goes a step further, creating even more distinct clusters that show its capability to capture the variety within the graph dataset, and the DiffPool method stands out with its highly distinct and compact clusters, which means that it is particularly good at understanding complex patterns. Our G-HIN2VEC method generates embeddings that are more distinct than Metapath2Vec but not as tightly clustered as HIN2VEC and DiffPool, suggesting that it provides a detailed, yet comprehensive overview of cardholder behavior. This aligns with the solid results we see from our quantitative analysis, where our method shows a strong ability to accurately group cardholder behaviors.

3.7 CONCLUSION

In this chapter, we introduced G-HIN2VEC, a novel approach for generating heterogeneous graph-level embeddings using a double-triplet loss in an unsupervised learning framework. This method enhances efficiency by avoiding graph-to-graph matching and sampling negatives from the entire dataset. Our evaluation on a real-world financial dataset, modeled as ego-centric graphs, demonstrated competitive performance, with notable improvements in gender classification accuracy and income prediction. Looking ahead, we plan to enhance G-HIN2VEC by integrating dynamic metapath selection, exploring advanced negative sampling techniques, and transitioning to an inductive learning model to improve generalization to unseen data.

4

Federated Learning Framework for Privacy-Preserving Anomaly Detection in Financial Transactions

CHAPTER CONTENTS

4.1	Introduction	81
4.2	Related Studies	83
4.3	Problem statement	84
4.4	Our approach	87
4.5	Design of experiments	96
4.6	Experiments and results	99
4.7	Conclusion	107

4.1 INTRODUCTION

Anomaly detection in financial transactions, particularly within the context of credit card transactions, is a critical task in combating fraud. According to the European Central Bank [Ban14], fraud in the Single Euro Payments Area (SEPA) reached 1.33 billion Euros in 2012 alone, marking a 14.8% increase compared to 2011. Furthermore, online fraud accounted for \$3.5 billion in losses in 2012, with an alarming 30% increase from 2010 [MB08]. Traditional anomaly detection models for fraudulent behavior often rely on attribute value data generated from transactional records, utilizing both supervised and unsupervised learning methods to identify suspicious activities [NHW⁺11a, SKSM08, WHJ⁺09].

Recently, graph-based methods have emerged as a powerful approach to identifying suspicious financial activities, by using both data attributes and graph topological information to detect anomalies [AMF10, TL11]. These methods have shown their effectiveness in leveraging both graph structure and attribute data to enhance anomaly and fraud detection in various applications. Comprehensive survey studies on anomaly detection [CBK09], including those using graph-based methods [ATK15, RSKH15, AKA17], and those specifically focused on fraud detection [BJTW11, AMZ16, NHW⁺11b], highlight the need to expand data collection to identify rare and unusual patterns or outliers within datasets that differ significantly across institutions.

Therefore, institutions dealing with the same anomaly and fraud detection challenges are increasingly interested in collaborative learning environments, where training occurs across distributed datasets without the need for raw data exchange, thereby safeguarding sensitive financial information [MMR⁺17, SZK⁺22], such in Federated Learning (FL) systems. Recent advancements in FL have demonstrated its potential to enhance fraud detection across multiple financial entities while maintaining the privacy of both individuals and institutions [YZY⁺19, ZYGW21]. However, the use of shared models in federated learning exposes institutions to significant risks, as highlighted in [LXW22]. In the case of tabular data, the nature of attribute values can lead to privacy leaks, especially when downstream tasks are trained to optimize cross-entropy-like objective functions [VBDV23, ZMB20, ZLH19].

In this chapter, we propose a novel federated learning framework for Privacy-Preserving Behavioral Anomaly Detection, leveraging Graph Neural Networks (GNNs) to model cardholder transactions as dynamic graphs. We utilize ego-centric graph modeling in [DSHS23a] as an **anonymization-based privacy mechanism** [Swe02] to eliminate the need for personally identifiable information (PII) in the training data, such as the cardholder’s unique identification number. Additionally, a **noise-based privacy mechanism** [CCG24] is employed to further protect sensitive data attributes. These privacy-preserving techniques are essential in mitigating the risk of Gradient Leakage [ZMB20], where gradients derived from sensitive financial data can reveal patterns related to individuals. By introducing noise, our framework ensures that even if the gradients are analyzed, sensitive patterns remain obfuscated, safeguarding the cardholder’s privacy. By utilizing dynamic graphs, this framework enables anomaly detection based on behavioral patterns rather than isolated events, providing a comprehensive understanding of fraudulent activities. By incorporating a **domain-specific negative sampling** technique, our framework effectively trains anomaly detection models without relying on labeled data, making it applicable to real-world scenarios where labeled datasets are unavailable.

Our main contributions are:

- We propose an end-to-end federated learning framework for privacy-preserving behavioral anomaly detection for credit card transactions, using dynamic graphs to model transaction patterns within a collaborative learning environment, enabling training across institutions without sharing sensitive data.
- We design a comprehensive privacy-preserving feature engineering, as a noised-based privacy mechanism, by encoding spatial and temporal financial attributes into embedding space while preserving individual privacy and critical characteristics like distance, orientation, and differentiation.
- We introduce a domain-specific negative sampling strategy to improve fraud detection by generating contextually relevant negative samples using dynamic graphs.
- We evaluate the effectiveness of the proposed framework on two benchmark datasets,

demonstrating its utility in improving detection accuracy while safeguarding individual privacy.

4.2 RELATED STUDIES

In this chapter, we focus on federated learning for anomaly detection in financial transactions using dynamic graph embedding over time, while addressing associated privacy concerns.

Graph-Level Embedding and Dynamic Graph Embedding.

Graph-level embedding techniques have gained traction for representing entire graphs as vectors, enabling tasks such as graph classification, similarity computation, and anomaly detection [YV15b, NCV⁺17]. Unlike node-level embeddings that preserve proximity between individual nodes, graph-level embeddings aim to capture the overall structural and attribute-based properties of entire graphs, preserving proximity between graphs. Traditional methods like Graph Kernels rely on counting subgraph occurrences, which can be computationally intensive and struggle with scalability [YV15b]. To overcome these limitations, newer methods like Graph2Vec have been developed to learn graph-level representations in an unsupervised manner [NCV⁺17].

Recent methods often utilize message passing algorithms, propagating information across the graph and aggregating it to update node representations. Graph Neural Networks (GNNs), such as Graph Convolutional Networks (GCNs) [KW17] and Graph Attention Networks (GATs) [VCC⁺17], exemplify this approach by learning comprehensive node representations from distant nodes across multiple layers, making them powerful tools for graph-level tasks. However, these methods primarily focus on static graphs.

Dynamic graphs are defined as graphs that evolve over time, with their structure changing due to the addition or removal of nodes and edges, or fluctuations in edge weights. Dynamic graph embedding methods extend traditional embedding techniques to capture graph temporal evolutions, which is crucial in domains like social networks, biological systems, and financial markets. These methods are designed to adapt to structural changes while maintaining the graph's temporal continuity and integrity [AS14, ANRD15]. For instance, the NetWalk approach efficiently handles large-scale streaming data by learning real-time repre-

sentations, making it particularly effective for anomaly detection in rapidly evolving cybersecurity networks [YCA⁺18]. Despite these advancements, embedding entire graphs over time while preserving both intra-graph and inter-graph proximity remains challenging. This work aims to adapt dynamic graph embeddings to detect suspicious behavior as a fraudulent transactions in financial systems, building on existing research in credit card fraud detection [BH01] and online fraud detection in auction networks [PCWF07].

Federated Learning (FL).

FL enables the collaborative training of models on distributed data [MMR⁺17], which is crucial for handling sensitive financial information. Recent studies have leveraged FL for fraud detection, focusing on supervised learning frameworks with labeled data [ASFEE24, ZYGW21]. These approaches demonstrate FL’s potential to improve fraud detection accuracy without requiring data sharing among institutions. Additionally, FL has shown effectiveness in detecting complex financial crimes across multiple entities, further reinforcing its significance in real-world financial applications [SZK⁺22]. However, existing federated fraud detection methods rely on training with raw data attributes, which exposes individual’s sensitive data to potential breaches in case of gradient leakage. Our approach seeks to address this issue by developing a framework that preserves the privacy of individuals, specifically cardholders, and participating institutions, by extending prior research on using GNN models to a privacy-preserving environment, introducing a noise-based privacy mechanism. Furthermore, the negative sampling strategy helps avoid label ambiguity and allows training without relying on labeled data, enhancing privacy and robustness.

4.3 PROBLEM STATEMENT

4.3.1 PRELIMINARIES

Local Differential Privacy. Differential Privacy (DP) [DMNS06] is a technique that ensures privacy by making it challenging for adversaries to extract meaningful information about individuals from a dataset. It prevents linkage attacks and ensures that even if an adversary knows an individual is in the dataset, they cannot determine specific information about that

individual.

Traditional DP, also known as Global Differential Privacy (GDP)[DMNSo6], relies on a trusted third party to manage privacy guarantees. However, finding such a trusted party is often impractical. Local Differential Privacy (LDP) addresses this by allowing individuals to perturb their raw data or model parameters before sharing it with a central server, ensuring that the server only receives anonymized data.

Definition (ϵ -LDP) [KLN⁺11]: For a randomized algorithm \mathcal{M} , let $D(\mathcal{M})$ and $R(\mathcal{M})$ denote its domain and range, respectively. For any two records I and I' in $D(\mathcal{M})$, consider a set of outputs $S \subseteq R(\mathcal{M})$ such that \mathcal{M} produces the same output S for both I and I' . The algorithm \mathcal{M} satisfies (ϵ, δ) -LDP if the following inequality holds:

$$\Pr[\mathcal{M}(I) = S] \leq e^\epsilon \cdot \Pr[\mathcal{M}(I') = S] + \delta,$$

where ϵ is the privacy budget, which determines the level of privacy protection, and δ represents the sensitivity of \mathcal{M} , indicating the tolerance for deviations from the expected privacy guarantee. In practice, a larger ϵ implies lower privacy protection for \mathcal{M} but potentially higher utility.

LDP is particularly valued for its properties of post-processing invariance and composability, which allow for robust privacy protection even as data undergoes multiple analyses.

Proposition 1 (Post-processing Invariance): If a randomized algorithm \mathcal{M}_1 satisfies ϵ -DP, then applying any further algorithm \mathcal{M}_2 to its output will also satisfy ϵ -DP.

Proposition 2 (Sequential Composition): If a series of algorithms \mathcal{M}_i each satisfy ϵ_i -DP, then their combined application to the same dataset will satisfy $\sum \epsilon_i$ -DP.

Personalized Local Differential Privacy. Local differential privacy has significant characteristics at individuals can independently disturb data to protect their privacy. As participants are the ones who understand their privacy requirements best, there may be varying

privacy requirements depending on the sensitivity of their data. Therefore, it is necessary to promote the concept of personalized privacy in local settings.

Definition (τ, ε) -PLDP[CLQ⁺16]: Assuming that \mathcal{M} is a randomized algorithm, given the personalized privacy parameters (τ, ε) of the participant/individuals and any pair of input values $I, I' \in \tau$, if and only if for any output $S \subseteq \text{Range}(\mathcal{M})$ of the randomized algorithm \mathcal{M} , the following inequality holds:

$$\Pr[\mathcal{M}(I) \in S] \leq e^\varepsilon \cdot \Pr[\mathcal{M}(I') \in S] + \delta,$$

Then the randomized algorithm \mathcal{M} is said to satisfy (τ, ε) -PLDP, where τ denotes the safe range, $\text{Range}(\mathcal{M})$ denotes the set of all possible outputs of \mathcal{M} , $\Pr[\cdot]$ denotes the probability, and ε denotes the privacy budget.

4.3.2 PROBLEM DEFINITION

FL enables clients to collaboratively train a global model by sharing gradients $g(x, y) = \nabla_{\theta} \mathcal{L}(f_{\theta}(x), y)$ with a central server. However, this process raises significant privacy concerns, particularly regarding the potential for data reconstruction, where the original data (x, y) could be recovered from the shared gradients [ZLH19, WL21].

The reconstruction problem, as simplified in [ZMB20], can be defined as:

$$\hat{x} = \arg \min_{x'} \mathcal{E}(g(x), g(x')),$$

where $\mathcal{E}(g(x), g(x'))$ represents the gradient matching loss minimized to reconstruct data x' by aligning it with the original shared gradients $g(x)$. This is achieved by iteratively updating the dummy data x' using gradient descent:

$$x' \leftarrow x' - \eta \nabla_{x'} \mathcal{L},$$

where \mathcal{L} is the loss function used in the gradient matching process, and η is the learning rate. Through multiple iterations, this process has the potential to reconstruct the original private data x from the shared gradients.

This chapter addresses two interrelated challenges within this context. First, we aim to mitigate the risk of data reconstruction to ensure the privacy of sensitive data, such as spatial and temporal attributes in financial transactions. This problem can be defined as the following divergence condition:

$$\mathcal{E}(g(x), g(x')) < \mathcal{E}(g(\tilde{x}), g(x')),$$

where $\tilde{x} = \mathcal{M}(x; \varepsilon, \delta)$ is generated by applying a mechanism \mathcal{M} that guarantees (ε, δ) -LDP, thereby reducing the risk of data reconstruction. Second, we focus on developing a federated learning framework that integrates privacy-preserving objectives while ensuring that the utility of individual data points is maintained for effective behavioral anomaly detection.

4.4 OUR APPROACH

In this section, we present our framework and the models illustrated in Figure 4.1.

4.4.1 GENERAL FRAMEWORK

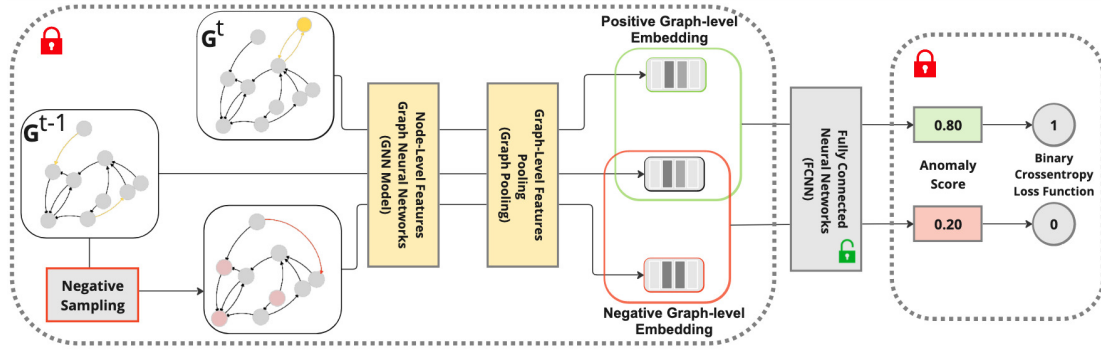


Figure 4.1: Federated anomaly detection with GNNs, sampling subgraphs, extracting features, and scoring anomalies.

The overview of the proposed framework is illustrated in Figure 4.1. At a high level, it comprises several key components: dynamic graph construction, negative sampling, node and graph-level feature extraction using GNNs, and anomaly detection via a fully connected neural network (FCNN). The framework operates within a federated learning setup, ensuring that sensitive data remains private while enabling collaborative model training across multiple participants.

Dynamic Graph Construction. We model the transaction data for each cardholder c_i as a dynamic graph $G_t^{(i)}$ at time t , defined as:

$$G_t^{(i)} = (V_t^{(i)}, E_t^{(i)}, X_t^{(i)}, T_t^{(i)}),$$

where the graph nodes $V_t^{(i)} = \{v_j\}$ are associated with attributes $(merchant_id, (lat_j, lon_j))$ transformed to features space denoted as $X_t^{(i)}$, where lat_j and lon_j denotes the location coordinates and $merchant_id$ denotes the merchant unique identifier. And the edges $E_t^{(i)} = \{e_{jk}\}$ represent transactions between consecutive merchants, where we consider valid sequences of consecutive transactions relevant when occurring within a two-hour rolling window*. An edge $e_{jk} \in E_t^{(i)}$ connects the nodes v_j and v_k , indicating a transaction timestamp from v_k af-

*Commonly used in [KRS⁺19, DCLoT⁺18]

ter v_j , denoted as $T_t^{(i)}$. The graph evolves over time as new transactions occur, capturing the dynamic nature of the cardholder’s transaction history.

The sequence of dynamic graphs of c_i can be denoted as $G^{(i)} = \{G_{t1}^{(i)}, G_{t2}^{(i)}, \dots, G_{tN}^{(i)}\}$ over time N to reflect the consecutive nature of transactions, where each dynamic graph $G_t^{(i)}$ builds upon a cumulative structure and transactions since the first active transactions, the same for the previous graph $G_{t-1}^{(i)}$. The graph collection $G^{(i)}$ effectively captures the temporal dependencies in the cardholder’s transaction behavior.

Negative Sampling. Inspired from [YCA⁺18], we train the proposed framework to distinguish between normal and anomalous patterns within cardholder dynamic graphs, we generate ”noisy” or perturbed graphs $\tilde{G}_t^{(i)}$ that deviate from typical transactional patterns of c_i observed in $G_{t-1}^{(i)}$. The generation process is denoted as:

$$\tilde{G}_t^{(i)} = \mathcal{N}\left(G_{t-1}^{(i)}\right)$$

where $\mathcal{N}(\cdot)$ applies domain-specific perturbations to $G_{t-1}^{(i)}$. Detailed steps are provided in Section 4.4.3.

Feature Extraction for Node and Graph Representations. At each time step t for a given cardholder c_i , the node features $H_t^{(i,0)}$, denoted for simplicity as $H_t^{(0)} \in \mathbb{R}^{|n_0| \times d}$, consist of feature engineered attributes, and the process is detailed in Section 4.4.2.

To extract higher-level features, a shared GNN architecture with L layers is initialized and deployed across all participants in the federation. At each layer l , the node embeddings $H_t^{(l)} \in \mathbb{R}^{|n_l| \times d_l}$ are updated through graph convolution operation [WPC⁺20] defined as follows:

$$H_t^{(l+1)} = \sigma\left(A_t^{(l)} H_t^{(l)} W^{(l)}\right),$$

where $W^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ are the trainable weight matrices, $A_t^{(l)} \in \mathbb{R}^{|n_l| \times |n_l|}$ is the adjacency matrix at time t encoding the graph structure and $\sigma(\cdot)$ represents the ReLU activation function.

To derive graph-level representations, a pooling mechanism is applied after each GNN layer,

progressively reducing the number of nodes by a factor of $p\%$. The number of nodes at layer $l+1$ is defined as $n_{l+1} = \lceil |V_t^{(l)}| \times p \rceil$. The pooling operation simultaneously updates the node feature matrix $H_t^{(l)}$ and the adjacency matrix $A_t^{(l)}$, resulting in a coarsened graph [LZL⁺22], we denoted by:

$$H_t^{(l+1)}, A_t^{(l+1)} = \text{Pooling}_{\lceil p\% \rceil} \left(H_t^{(l)}, A_t^{(l)} \right).$$

This iterative process continues until the final layer, where all nodes are pooled into a single cluster, producing a final embedding vector $q_t^{(i)}$ that represents the entire graph. This graph-level representation encapsulates the transaction behavior of the cardholder c_i at time t .

In certain GNN architectures, the pooling layer is trained [GJ19, KTA19, LLK19, RST20, YYM⁺18], eliminating the need for a predefined parameter $p\%$. Similarly, for the graph convolution operation, alternative approaches such as message-passing frameworks [GSR⁺17, LTBZ16] can be used within the proposed framework.

Discriminative Behavioral Anomaly Detection with FCNN. The final graph embedding $q_{t-1}^{(i)}$ and $q_t^{(i)}$ serves as input to a fully connected neural network (FCNN) that acts as a discriminative model for behavioral anomaly detection, denoted as $f: \mathbb{R}^d \rightarrow [0, 1]$. The FCNN consists of multiple dense layers (MLP) with ReLU activation functions, followed by a softmax layer that outputs the anomaly score. The model is trained end-to-end using a cross-entropy loss function:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)],$$

where y_i represents the true label, and \hat{y}_i is the predicted probability for the i -th generated sample, where $f(q_{t-1}^{(i)}, q_t^{(i)}) \rightarrow 1$ and $f(q_{t-1}^{(i)}, \tilde{q}_t^{(i)}) \rightarrow 0$, where

$$\text{Anomaly Score} = f(q_{t-1}^{(i)}, q_t^{(i)}), \quad i \in \{1, 2, \dots, K\}$$

In our federated learning setting, each client generates a finite number of samples N and computes gradients based on their local model updates and shares only these gradients with the central server. The server aggregates these gradients to update the global model, ensuring that the learned representations are robust and privacy-preserving.

4.4.2 FEATURE ENGINEERING WITH PRIVACY PRESERVATION

Inspired by the position encoding architecture in Transformer [VSP⁺17], we feature engineer spatial (location), temporal (timestamp) attributes in financial data as positional embeddings. To address the data breach issue [VBDV23], we propose transforming low-dimensional features with privacy-preserving properties into high-dimensional representations of size d as the embedding size, which are then fed into local models, where a noise-based privacy mechanism is applied, as detailed in Algorithm 3.

Location. Building on the foundation of multi-scale location encoders [Mea20, Zea20], we adapt this approach to be unbounded, as in a FL environment, the minimum and maximum grid scales could compromise the privacy of individual participants.

Given latitude φ and longitude λ in degrees, the coordinates are first converted to radians, denoted as φ_{rad} and λ_{rad} . To generate the high-dimensional representation, we define the scaling factors \mathbf{s}_φ and \mathbf{s}_λ as follows:

$$\mathbf{s}_\varphi = \frac{2\pi\varphi_{\text{max}}}{C_{\text{earth}}} \cdot \frac{1}{2^k}, \quad \mathbf{s}_\lambda = \frac{2\pi}{2^k}, \quad k \in \{1, 2, \dots, d\}.$$

The latitude \mathbf{l}_φ and longitude \mathbf{l}_λ vectors are defined as:

$$\begin{aligned} \mathbf{l}_\varphi &= [\sin(\varphi_{\text{rad}}) \cdot \mathbf{s}_\varphi, \cos(\varphi_{\text{rad}}) \cdot \mathbf{s}_\varphi]^\top, \\ \mathbf{l}_\lambda &= [\sin(\lambda_{\text{rad}}) \cdot \mathbf{s}_\lambda, \cos(\lambda_{\text{rad}}) \cdot \mathbf{s}_\lambda]^\top. \end{aligned}$$

Contrary to [Mea20], the final location representation is obtained by summing the latitude

and longitude vectors

$$\mathbf{l}_{\text{location}} = \sigma(\mathbf{l}_\varphi + \mathbf{l}_\lambda).$$

Here, σ is a non-linear function. The final vector representation preserves both the directional and distance properties, as empirically demonstrated in Section 4.6.4.

Temporal. To minimize the risk of compromising FL while handling temporal features such as using one-hot encoding [VBDV23], we transform time features into a dense and continuous space.

Given a timestamp, the temporal data is embedded in a high-dimensional vector space to capture cyclical patterns at various granularities decomposed into hours, minutes, seconds, days, and months.

To generate the high-dimensional representation, we define the scaling factor \mathbf{s}_k for each dimension k as:

$$\mathbf{s}_k = \frac{1}{k} \sin\left(\frac{2\pi k}{d}\right), \quad k \in \{1, 2, \dots, \frac{d}{2}\}$$

For each temporal component $\lambda \in \{\text{hour, minute, second, day, month}\}$, with periodicity N_λ , the vector representation is defined as:

$$\tau_\lambda = \left[\sin\left(2\pi \frac{\lambda}{N_\lambda}\right) \cdot \mathbf{s}_k, \cos\left(2\pi \frac{\lambda}{N_\lambda}\right) \cdot \mathbf{s}_k \right]^\top$$

The final temporal representation is obtained by summing all granularities vectors:

$$\tau_{\text{temporal}} = \sigma\left(\sum_{\lambda} \tau_\lambda\right)$$

This vectorial representation effectively captures the cyclical nature of time across different

temporal scales, enabling robust temporal modeling, as empirically demonstrated in Section 4.6.4.

Merchant ID. Given a set of unique merchant IDs V in the entire dataset, each merchant ID is represented as a dense vector of size d , stored in a matrix $\mathbf{M} \in \mathbb{R}^{|V| \times d}$, where d is the dimensionality of the embedding. For a given merchant ID $m \in V$, the corresponding row vector \mathbf{m}_m is selected from \mathbf{M} , making \mathbf{M} a merchant lookup table.

In this section, the embedding matrix \mathbf{M} is initialized randomly using normal distributions with standard deviations of 0.1. These embeddings are then fine-tuned during model training rounds to capture merchant-specific patterns in transaction data across the federation.

Algorithm 3: $(\varepsilon_i, \delta_i)$ -Private Shuffle for Feature Vector

```

1: Input: Feature vector  $v = (v_1, \dots, v_d) \in [-1, 1]^d$ , privacy budget
    $\varepsilon = (\varepsilon_1, \dots, \varepsilon_d) > 0$ , privacy parameter  $\delta = (\delta_1, \dots, \delta_d) \geq 0$ , random seed  $s$ 
2: Output: Perturbed and shuffled vector  $\tilde{x} \in [-1, 1]^d$ 
3: Set random seed  $s$ 
4: Random permutation  $r : [d] \leftarrow [d]_s$ 
5: for  $j \in [d]$  do
6:    $\tilde{x}_j \leftarrow x_j + \mathcal{N}(0, \frac{\delta_{r(j)}}{\varepsilon_{r(j)}})$ 
7:    $\tilde{x}_j \leftarrow \min(\max(\tilde{x}_j, 1), -1)$ 
8: end for
9:  $\tilde{x} \xleftarrow{r} \tilde{x}$ 
10: return  $\tilde{x}$ 

```

Privacy-Preserving Feature Engineering. Building upon the generalized differential privacy framework presented in [CCG24], we extended the application of the privacy preservation to encoded financial data attributes—location (L_{location}), temporal (T_{temporal}), and merchant ID (M)—through a personalized differential privacy mechanism coupled with a vectorial shuffling scheme to generate $L'_{\text{location}}, T'_{\text{temporal}}, M'$ as noise-based privacy mechanism for privacy-preserving feature matrices, as detailed in Algorithm 3.

Following [YCA⁺18], we apply a $(\varepsilon_i, \delta_i)$ -Private Shuffle to perturb feature vectors before their use in our federated model, as described in Algorithm 1. The algorithm begins by ran-

domly permuting the elements of a feature vector v using a fixed random seed. Each element is then perturbed with noise introduced by the Gaussian mechanism [DMNS06], where the noise level is determined by specific privacy parameters ε_i and δ_i assigned to each feature. This personalized perturbation allows for differential control over privacy budgets across features.

Subsequently, the algorithm clips the perturbed vector elements to the range $[-1, 1]$ to maintain feature integrity. The final step involves shuffling the vector to further obscure the original feature order, enhancing privacy as proved in [MCCJ22, CCG24].

Notably, as demonstrated in [CCG24], Algorithm 1 approximately preserves μ -Gaussian Differential Privacy (μ -GDP), where:

$$\mu = \sqrt{\frac{2}{\sum_{i=1}^n \frac{1-\delta_i}{1+e^{\varepsilon_i}} - \max_i \frac{1-\delta_i}{1+e^{\varepsilon_i}}}}.$$

For a given cardholder c_i , the initial node features \mathbf{X}_i are constructed by integrating location information $\mathbf{L}'_i \subset L'$ and merchant ID attributes $\mathbf{M}'_i \subset M'$, expressed as $\mathbf{X}_i = \mathbf{L}'_i + \mathbf{M}'_i$. This element-wise addition captures the combined spatial and merchant characteristics of each node.

Temporal information $\mathbf{T}'_i \subset T'$ is then incorporated into the message-passing process, where φ is a simple element-wise addition. The normalized features are updated using:

$$\mathbf{H}_i^{(0)} = \varphi \left(\mathbf{X}_i, \bigoplus_{j \in \mathcal{N}(i)} \tanh(\mathbf{X}_j + \mathbf{T}'_{ij}) \right)$$

where $\mathcal{N}(i)$ denotes the set of neighbors for node i . This approach effectively captures interactions by leveraging both spatial and temporal information, enabling the input feature space to incorporate complex patterns in graph-structured data.

4.4.3 NEGATIVE SAMPLING WITH GRAPH CONFIGURATION FOR MODEL TRAINING

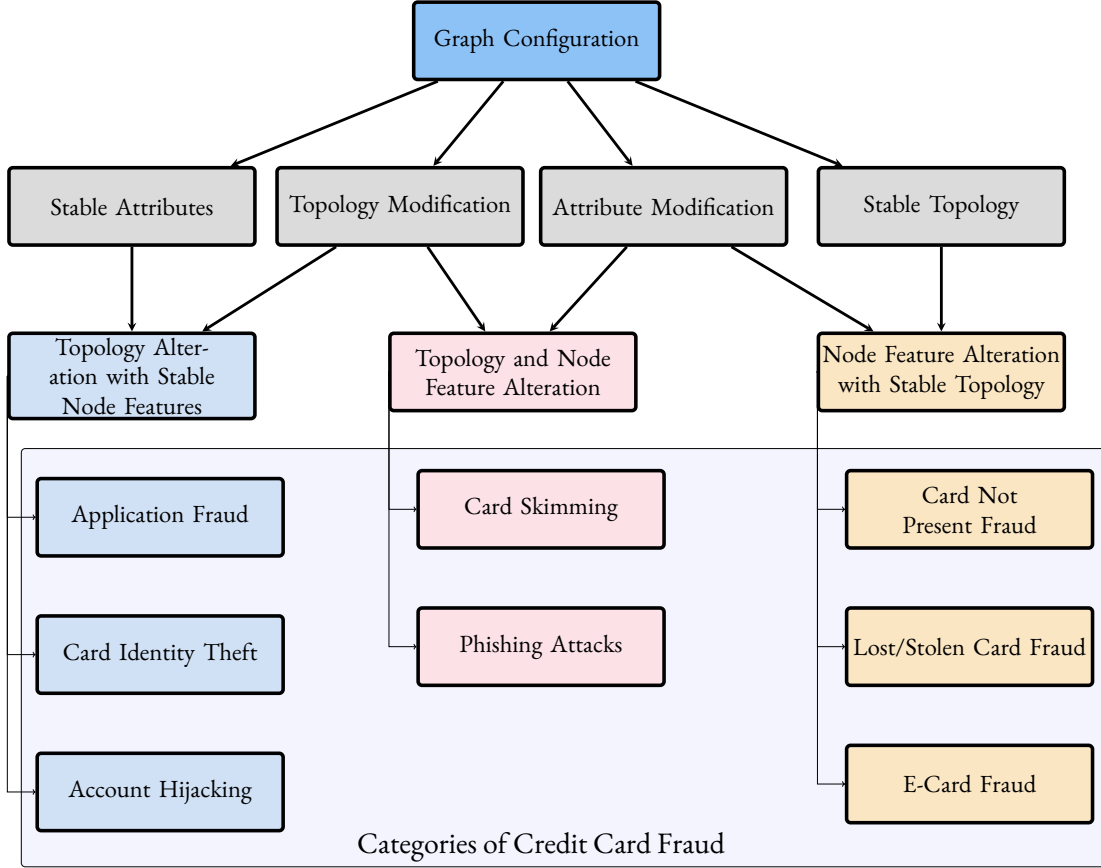


Figure 4.2: Categorization of credit card fraud types based on graph configuration manipulations, illustrating how different alterations in topology and node features correspond to specific fraud scenarios.[BPD03]

To effectively train the federated model, we introduce a domain-specific negative sampling strategy that leverages graph configuration perturbations based on various types of credit card fraud, illustrated in Figure 4.2. Rather than generating negative samples from unrelated cardholder graphs, our approach directly perturbs the graph $G_t^{(i)}$, resulting in contextually relevant and computationally efficient negative local instances.

For each graph, a graph configuration $\mathcal{C} \in \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$ is randomly selected from a pre-defined set, with each configuration designed to simulate a specific type of fraud attack by altering the topology, node features, or both. The configurations are defined as follows:

- Topology Alteration with Stable Node Features (\mathcal{C}_1):

$$\tilde{A}_t^{(i)}, \tilde{E}_t^{(i)} = \mathcal{F}_T(A_t^{(i)}, E_t^{(i)}), \quad \tilde{X}_t^{(i)} = X_t^{(i)}$$

- Node Feature Alteration with Stable Topology (\mathcal{C}_2):

$$\tilde{A}_t^{(i)} = A_t^{(i)}, \quad \tilde{E}_t^{(i)} = E_t^{(i)}, \quad \tilde{X}_t^{(i)} = \mathcal{F}_V(X_t^{(i)})$$

- Topology and Node Feature Alteration (\mathcal{C}_3):

$$\tilde{A}_t^{(i)}, \tilde{E}_t^{(i)} = \mathcal{F}_T(A_t^{(i)}, E_t^{(i)}), \quad \tilde{X}_t^{(i)} = \mathcal{F}_V(X_t^{(i)})$$

Here, \mathcal{F}_T represents the randomization process applied to the adjacency matrix and edge set, perturbing the graph's topology. Meanwhile, \mathcal{F}_V involves node injection and swapping, where $\tilde{X}_t^{(i)} \subset X_t$, selectively altering node features within the graph to simulate more complex fraud patterns. These perturbations result in domain-specific negative samples $\tilde{G}_t^{(i)}$ that remain contextually relevant to the cardholder's transactional behavior.

4.5 DESIGN OF EXPERIMENTS

To evaluate the proposed framework, we use two public financial datasets and simulate a federated environment with multiple clients. Each dataset is divided into five subsets to create local datasets uniformly distributed by cardholder ID, resulting in varying data sizes due to individual cardholder total transactions. Each local dataset is split by cardholder ID, with 75% for training and 25% for testing. To compare graph pooling methods for cardholder behavior representation under privacy protection, we use the F1 score as the primary metric, as it balances precision and recall, particularly in imbalanced scenarios. Additionally, we report Accuracy, Precision, Recall, and ROC-AUC for a comprehensive performance evaluation. All metrics are averaged across all test sets of the federation participants.

4.5.1 PARAMETER SETTINGS

The model parameters were tuned using 5-fold cross-validation to ensure optimal performance. For the GNN model, we configured $L = 5$ layers with a node and edge embedding dimension of $d = 128$, the same dimension used for all data attributes in the vectorial representation. The weight matrices were initialized using Xavier initialization [GB10], and ReLU was used as the activation function throughout. We applied a 5% pooling factor at each layer, with a dropout rate of 0.1 to prevent overfitting. The FCNN consisted of 2 dense layers, using ReLU activations with a Softmax function in the final layer. The Adam optimizer [Kin14], with a learning rate of 0.001, was used for training. In the FL framework, 5 clients were employed, each training locally for 10 epochs, with 200 communication rounds. The FedProx aggregation method [LSZ⁺20] is used to merge gradient over all training rounds. For the personalized DP mechanism [LMW⁺20], the privacy budget ϵ was uniformly set between 0.01 and 10, with a constant privacy parameter $\delta = 10^{-4}$, and a fixed shuffling seed was maintained across all clients.

4.5.2 DATASETS

We employ two financial datasets from different regions and economic contexts: one from a Brazilian bank (ELO BR)[†] and one from a United States bank (IBM US)[‡]. The datasets are tabular with the following schema fields: `cardholder_id` (as INT), `merchant_id` (as INT), `timestamp` (as Timestamp), `amount` (as NUMERIC), `lat` (as DOUBLE), `lon` (as DOUBLE), and `MCC` (as VARCHAR), which stands for Merchant Category Code[§], offering detailed insights into credit card transactions. All data types are denoted according to SQL data types.

The original datasets comprise 327,541 merchants from Brazil (ELO BR) and 99,554 merchants from the United States (IBM US), along with 322,862 customers (ELO BR) and 4,967 customers (IBM US). In line with previous studies [KRS⁺19, DCLOT⁺18], we pre-

[†]Kaggle | Elo Merchant Category Recommendation: [link](#).

[‡]Synthetic Transaction Dataset: [link](#).

[§]MCC refers to a standardized code system used by card networks.

processed the datasets by filtering out inactive cardholders with less than 3 transactions per month to concentrate on more significant transaction patterns.

4.5.3 BASELINES

We compare our proposed framework against state-of-the-art baselines, which can be categorized into two groups: deep learning-based methods and clustering-based graph pooling methods.

- **TopKPooling** [GJ19, KTA19] is an advanced graph pooling technique that selects the top K nodes based on a learned scoring function. This method preserves the most informative nodes while reducing the graph's complexity.
- **SAGPooling** [LLK19, KTA19] is a self-attention based graph pooling method that assigns attention scores to nodes and aggregates them based on these scores. It allows the model to focus on the most relevant parts of the graph for a downstream task.
- **ASAPooling** [RST20] is an adaptive structure-aware pooling method that captures local substructures by clustering nodes before pooling, thus preserving important local information while reducing the graph size.
- **DiffPool** [YYM⁺18] is a hierarchical graph pooling method that learns a differentiable soft assignment of nodes to clusters, enabling the creation of a coarser graph while preserving its hierarchical structure. This approach is particularly effective in capturing global graph properties.

As a clustering-based graph pooling method, this approach first applies a clustering algorithm K-Means, Agglomerative Clustering, and Spectral Clustering at the node level with k clusters. Following the clustering step:

- **Max-Pooling** selects the maximum value within each node's features, preserving the most prominent features within each cluster to generate a concise graph representation [Mem24, Ano22b, Ano22a].

- **Avg-Pooling** computes the average node features within each cluster, yielding a smooth and generalized graph representation [Ano22a, Ano22b].

4.6 EXPERIMENTS AND RESULTS

We present privacy-preserving experimentation results on federated classification tasks for anomaly detection and fraud prevention, using real and synthetic payments data

4.6.1 PRIVACY BUDGET AND MODEL UTILITY

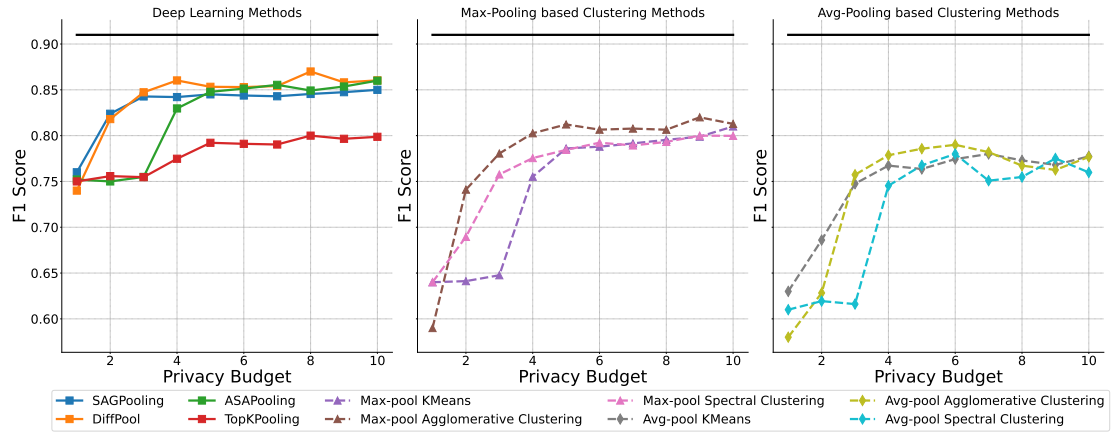


Figure 4.3: ELO Data: Impact of Privacy Budget on F1 Score Across Various Graph Pooling Methods.

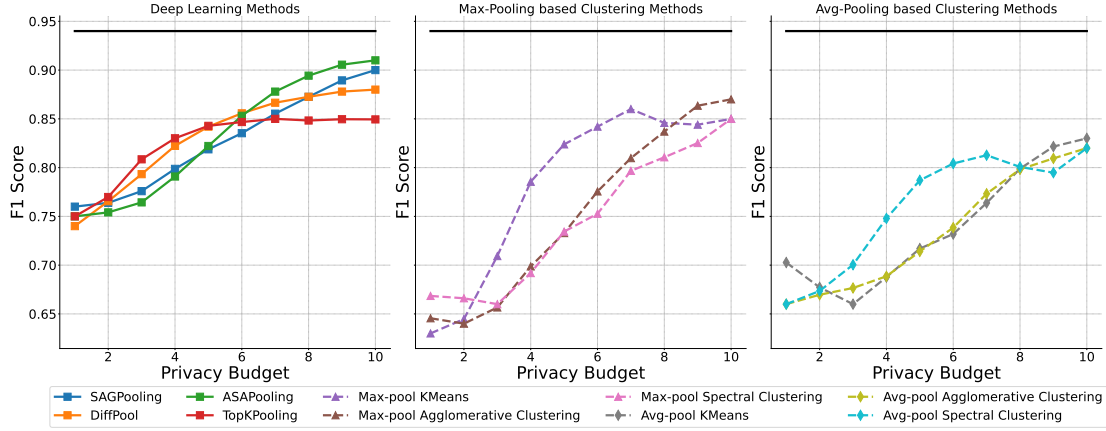


Figure 4.4: Synthetic Data: Impact of Privacy Budget on F1 Score Across Various Graph Pooling Methods.

In the synthetic dataset, deep learning methods achieve significant improvements in model performance as the privacy budget increases. At $\epsilon = 2$, the F1 score starts at 0.7 and progressively improves, reaching 0.95 at $\epsilon = 10$, closely approaching the centralized model's performance. This improvement occurs as the noise applied to the features decreases, allowing for better data representation. A similar trend is observed in the ELO dataset, where F1 scores rise from 0.65 at $\epsilon = 2$ to 0.9 at $\epsilon = 10$. The results highlight that higher privacy budgets lead to enhanced feature utility, which was expected, particularly that the attribute features are sensitive to any noising mechanisms. Although a privacy budget of 10 is generally considered high, we consider it acceptable due to the sensitivity of the data attributes and the application of privacy mechanisms at the cardholder level. To maximize model performance while maintaining privacy, ϵ is uniformly sampled from a range of 0.01 to 10 for each cardholder, ensuring personalized privacy protection following [LMW⁺20].

In contrast to earlier findings, clustering-based methods show gains with increasing privacy budgets, but even at the highest privacy budget, their improvements are modest compared to deep learning methods. In the synthetic dataset, Max-Pooling methods stabilize at an F1 score of 0.85, while Avg-Pooling methods reach 0.8 by $\epsilon = 10$. On the ELO dataset, these methods demonstrate similar behavior, but neither fully converges with the centralized model's performance. These observations suggest that while all models benefit from increased privacy budgets, deep learning methods are more effective in utilizing the additional

data fidelity. The consistent results across both datasets demonstrate the robustness of the proposed framework in providing better model utility and privacy trade-offs in a federated learning environment.

4.6.2 ANOMALY DETECTION RESULTS

Dataset	Pooling Model	Clustering Algorithm	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Synthetic Data	Centralized Model		0.98 ± 0.02	0.96 ± 0.02	0.94 ± 0.03	0.95 ± 0.02	0.97 ± 0.02
	Max-pool	KMeans	0.88 ± 0.04	0.87 ± 0.04	0.86 ± 0.05	0.86 ± 0.04	0.90 ± 0.04
		Agglomerative Clustering	0.89 ± 0.04	0.88 ± 0.04	0.87 ± 0.05	0.87 ± 0.04	0.91 ± 0.04
		Spectral Clustering	0.88 ± 0.05	0.85 ± 0.05	0.85 ± 0.05	0.85 ± 0.05	0.89 ± 0.05
	Avg-pool	KMeans	0.86 ± 0.05	0.84 ± 0.05	0.83 ± 0.06	0.83 ± 0.05	0.88 ± 0.05
		Agglomerative Clustering	0.87 ± 0.05	0.85 ± 0.05	0.84 ± 0.06	0.84 ± 0.05	0.89 ± 0.05
		Spectral Clustering	0.86 ± 0.06	0.83 ± 0.06	0.82 ± 0.06	0.82 ± 0.06	0.87 ± 0.06
	TopKPooling	---	0.89 ± 0.03	0.86 ± 0.03	0.85 ± 0.04	0.85 ± 0.03	0.91 ± 0.03
	SAGPooling	---	0.93 ± 0.02	0.91 ± 0.02	0.90 ± 0.02	0.90 ± 0.02	0.95 ± 0.02
	ASAPooling	---	0.94 ± 0.02	0.92 ± 0.02	0.91 ± 0.02	0.91 ± 0.02	0.96 ± 0.02
	DiffPool	---	0.91 ± 0.03	0.89 ± 0.03	0.92 ± 0.03	0.88 ± 0.03	0.94 ± 0.03
Brazilian Bank	Centralized Model		0.94 ± 0.02	0.91 ± 0.02	0.93 ± 0.03	0.92 ± 0.02	0.92 ± 0.02
	Max-pool	KMeans	0.83 ± 0.05	0.82 ± 0.05	0.81 ± 0.06	0.81 ± 0.05	0.85 ± 0.05
		Agglomerative Clustering	0.84 ± 0.05	0.83 ± 0.05	0.82 ± 0.06	0.82 ± 0.05	0.86 ± 0.05
		Spectral Clustering	0.83 ± 0.06	0.80 ± 0.06	0.79 ± 0.06	0.80 ± 0.06	0.84 ± 0.06
	Avg-pool	KMeans	0.81 ± 0.06	0.79 ± 0.06	0.78 ± 0.07	0.78 ± 0.06	0.82 ± 0.06
		Agglomerative Clustering	0.82 ± 0.06	0.80 ± 0.06	0.79 ± 0.07	0.79 ± 0.06	0.83 ± 0.06
		Spectral Clustering	0.81 ± 0.07	0.78 ± 0.07	0.77 ± 0.07	0.78 ± 0.07	0.82 ± 0.07
	TopKPooling	---	0.84 ± 0.04	0.81 ± 0.04	0.80 ± 0.05	0.80 ± 0.04	0.85 ± 0.04
	SAGPooling	---	0.88 ± 0.03	0.86 ± 0.03	0.85 ± 0.04	0.85 ± 0.03	0.89 ± 0.03
	ASAPooling	---	0.89 ± 0.03	0.87 ± 0.03	0.86 ± 0.04	0.86 ± 0.03	0.90 ± 0.03
	DiffPool	---	0.86 ± 0.04	0.84 ± 0.04	0.87 ± 0.04	0.87 ± 0.04	0.87 ± 0.04

Table 4.1: Performance of Different Graph Pooling Methods for Federated Anomaly Detection (Including Centralized Model).

In this subsection, the reported performance of various graph pooling methods is under a high privacy budget $\epsilon = 10$. The results in Table 1 demonstrate that GNN-based pooling methods, such as ASAPooling and DiffPool, consistently outperformed traditional techniques. ASAPooling, in particular, achieved impressive metrics, with a ROC-AUC of 0.96 ± 0.02 , accuracy of 0.94 ± 0.02 , and F1 score of 0.91 ± 0.02 across all test sets. This highlights the ASAPooling’s capability to maintain data privacy while preserving critical subgraph

structures, essential in federated settings. DiffPool also performed strongly, especially in recall, with an average score of 0.92 ± 0.03 , indicating effectiveness in detecting a wide range of anomalous behaviors generated through the negative sampling strategy 4.4.3, particularly those influenced by temporal patterns in transactional data.

The success of GNN-based methods can be largely attributed to their use of message passing and graph convolution for feature extraction, which is crucial for the performance of anomaly detection models.

Clustering-based methods and traditional graph pooling techniques, such as Max-pool and Avg-pool, showed limited performance gains, even with an increased privacy budget. The minimal improvements suggest inherent limitations in these methods ability to generalize across diverse and distributed datasets, particularly since clustering algorithms were not federated and operated independently and both the Max and Avg pooling are static graph pooling layers. Max-pool achieved an average F1 score of 0.85, while Avg-pool reached only 0.80, indicating that both approaches struggle with the complexity of federated graph data. These findings point to the need for more advanced models, like federated clustering algorithms, which could better manage privacy constraints and improve feature extraction at the FCNN level.

Overall, the findings strongly support the effectiveness of GNN-based pooling methods within federated learning frameworks, particularly under privacy requirements. Both ASAPooling and DiffPool consistently demonstrated high performance and adaptability, effectively overcoming the challenges associated with federated environments.

4.6.3 APPLICATION TO FRAUD DETECTION

Dataset	Client	Local F _I	Local+AS $\Delta\%$ in F _I	FL $\Delta\%$ in F _I	FL+AS $\Delta\%$ in F _I	Diff ($\Delta\%$)
Synthetic Data	1	0.81	+2.0%	+3.6%	+5.1%	+1.5%
	2	0.83	+2.3%	+3.0%	+4.6%	+1.6%
	3	0.82	+2.1%	+3.3%	+5.5%	+2.2%
	4	0.80	+2.5%	+3.9%	+6.1%	+2.2%
	5	0.82	+2.4%	+3.8%	+5.6%	+1.8%
Brazilian Bank	1	0.73	+1.9%	+5.0%	+7.8%	+2.8%
	2	0.71	+2.1%	+5.5%	+7.4%	+1.9%
	3	0.74	+1.6%	+6.3%	+7.0%	+0.7%
	4	0.70	+2.0%	+5.7%	+7.2%	+1.5%
	5	0.72	+1.8%	+6.1%	+7.5%	+1.4%

Table 4.2: Impact of Federated Anomaly Score (AS) Integration on Federated Fraud Detection Performance, Measured by F1 Score Improvements.

We evaluated the integration of the Federated Anomaly Score (AS) derived from our proposed anomaly detection framework, which is backed by DiffPool GNN-based methods, into a federated fraud detection model. The results, as shown in Table 4.2, demonstrate a significant improvement in the F_I scores when AS is used as an input feature. For the synthetic dataset, incorporating AS led to an increase in F_I score of 1.5% to 2.2% between all clients, while the Brazilian Bank dataset showed gains ranging from 0.7% to 2.8%. These enhancements underscore the effectiveness of the AS, particularly due to its ability to capture diverse transactional anomalous patterns through DiffPool.

Overall, the integration of the anomaly score within the federated learning framework significantly boosts fraud detection accuracy while preserving privacy. The consistent performance improvements across different datasets and clients highlight the robustness and applicability of the proposed method in real-world financial environments, where maintaining data pri-

vacy is crucial.

4.6.4 PERSONALIZED PRIVATE FEATURES

		Pearson	Spearman	Kendall	dCov[SRB07]
Geodesic Distance	Location Representation				
	Bearing Angle	0.8349	0.6107	0.4337	0.7190
	Euclidean Distance	0.9509	0.9148	0.7814	0.9452
	Cosine Distance	0.9150	0.9148	0.7814	0.9291
	Manhattan Distance	0.9549	0.9116	0.7711	0.9435
	Minkowski Distance	0.9450	0.9147	0.7810	0.9437
	Chebyshev Distance	0.9359	0.9144	0.7801	0.9416
Time Difference	Temporal Representation				
	Euclidean Distance	0.6294	0.5898	0.4289	0.5932
	Cosine Distance	0.5000	0.4901	0.3398	0.4818
	Manhattan Distance	0.6305	0.6137	0.4323	0.5905
	Minkowski Distance	0.6318	0.5823	0.4359	0.5997
	Chebyshev Distance	0.6325	0.5716	0.4514	0.6044

Table 4.3: Correlation Analysis of Location and Temporal Vectorial Representations. The table summarizes the preservation of distance, orientation, and time relationships using different correlation methods.

A critical aspect of our framework is the integration of personalized private features within the federated learning model. Through simulations involving 100,000 data pairs, we evaluated the correlation between the privacy-preserving embeddings and real-world metrics. The effectiveness of feature privatization was assessed under a controlled simulation environment, focusing on the relationship between actual values and their embedding-based representations.

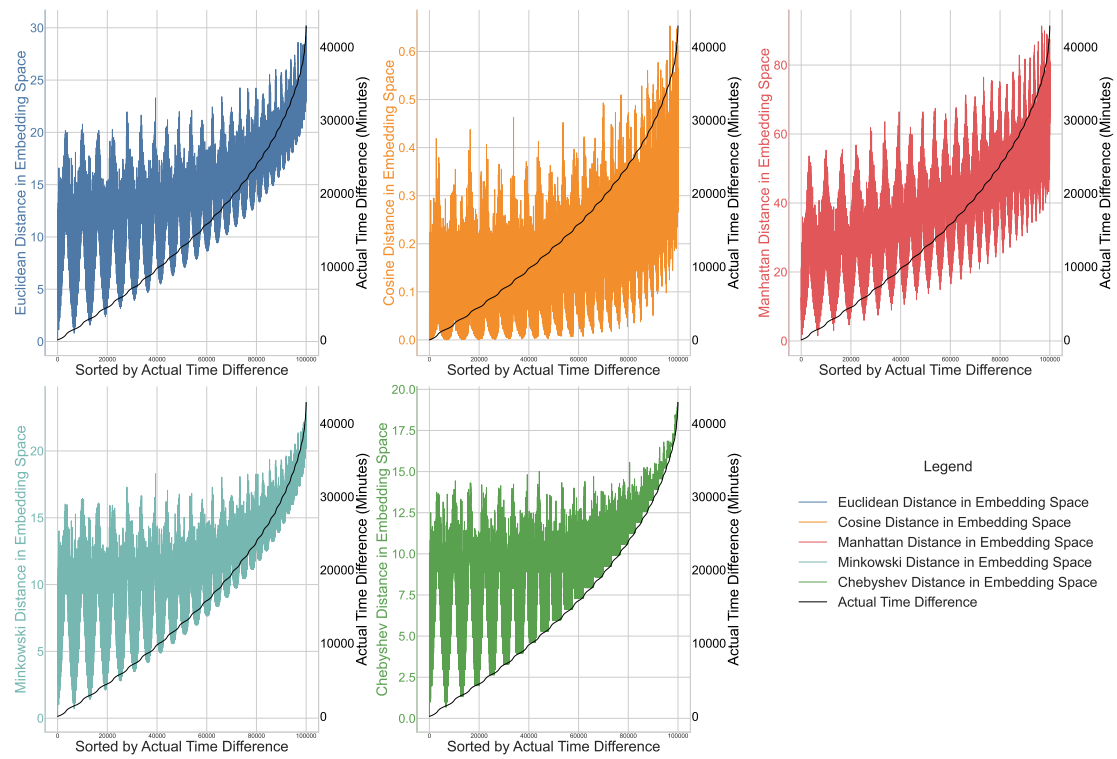


Figure 4.5: Temporal Embedding Distance

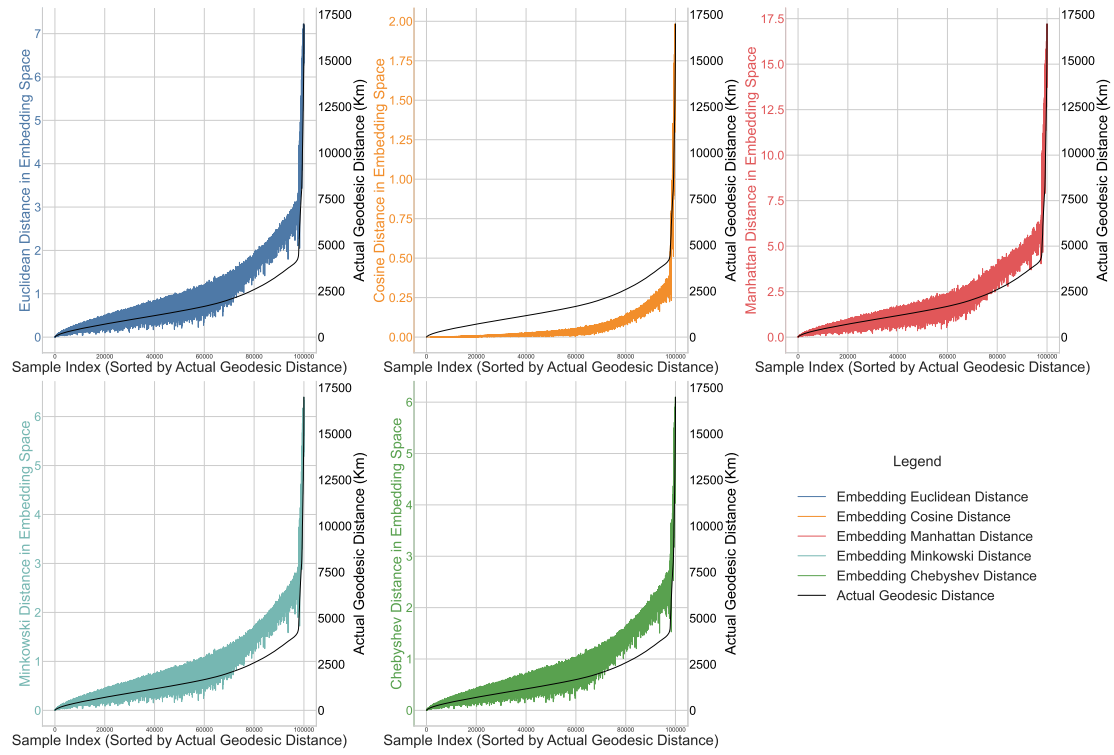


Figure 4.6: Spatial Embedding Distance

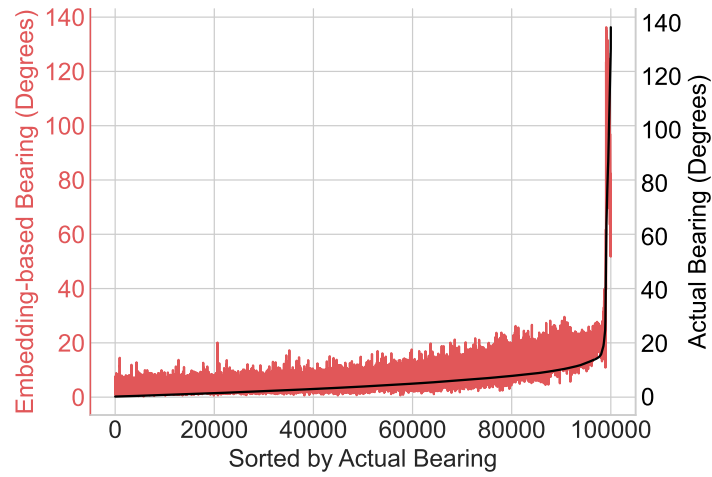


Figure 4.7: Bearing Embedding Distance

The correlation analysis is conducted under a personalized differential privacy mechanism, detailed in 3, with a privacy budget ϵ set between 0.01 and 1 and a constant privacy parameter $\delta = 10^{-4}$, revealed that our embeddings maintain strong spatial and temporal relationships despite privacy constraints. Given the sensitive nature of the data attributes involved, these findings underscore the importance of carefully selecting privacy budgets and their distributions, as they can significantly impact both the accuracy and stability of the proposed framework.

For **location-based features**, geodesic distance and bearing angle exhibited high correlations across various metrics, with Pearson correlations of 0.95 and 0.83, respectively. These findings indicate that location embeddings effectively preserve spatial distances and directional orientation, essential for accurate geospatial modeling. The strong linear trends observed in the geodesic distance and bearing angle figures 4.7 and 4.6 further support this conclusion.

Temporal features showed moderate correlations, with a Pearson correlation of 0.62 for time differences. While these embeddings capture the cyclical nature of time, the variability in the time difference figure 4.6 suggests some limitations in representing finer temporal details. Nonetheless, the correlations are sufficient to support robust temporal modeling in privacy-sensitive applications.

These results highlight the robustness of our privacy-preserving feature engineering approach, demonstrating that even under stringent privacy constraints, the embeddings retain critical characteristics of the original data. This emphasizes the need for precise calibration of privacy parameters to ensure optimal model performance in federated learning environments.

4.7 CONCLUSION

In this chapter, we presented a federated learning framework for privacy-preserving behavioral anomaly detection in financial transactions. Utilizing Graph Neural Networks on dynamic graphs, our approach captures the evolving nature of transactions while preserving cardholder privacy. By integrating a novel negative sampling technique, we effectively trained anomaly detection models without labeled data. Our experimental results, using datasets

from both a US bank and a Brazilian bank, demonstrated that deep learning-based methods, particularly ASAPooling and DiffPool, outperformed traditional clustering-based approaches, with significant improvements in fraud detection accuracy.

5

Federated Byte-level BPE Tokenizer: A Privacy-Preserving Approach for Distributed Language Tokenization

CHAPTER CONTENTS

5.1	Introduction	111
5.2	Related Studies	112
5.3	Problem statement	114
5.4	Our approach	116
5.5	Design of experiments	124
5.6	Experiments and results	127
5.7	Conclusion	135

5.1 INTRODUCTION

Tokenizers are tools or algorithms used in Natural Language Processing (NLP) to convert text into an organized structure, typically by breaking it down into smaller units known as tokens, which can be words, subwords, characters, or bytes. These tools are fundamental to most state-of-the-art Language Models (LMs) utilized in various downstream NLP tasks, and are mainly trained on a centralized dataset. However, this data-centric LM training approach faces challenges when individual data, especially sensitive and personally identifiable information (PII), cannot be centralized or exposed for processing in industries such as healthcare and finance in a cross-silo setting. Federated learning [MRTZ17] presents a novel solution by facilitating collaborative model training across multiple clients without the need for direct data sharing, thus offering privacy-preserving guarantees [DR⁺14]. This method, which contrasts with traditional LM training, allows clients to train local models with initial parameters from a (trusted) central server and only share back the updated local parameters for global aggregation. These become the new global parameters for all client models. After several rounds, the aggregator server shares the final model parameters. This paradigm adheres to regulations like the General Data Protection Regulation (GDPR) in the European Union by supporting collaborative learning without exposing sensitive individual information, such as bank transactions, geographical locations, and textual communications.

Privacy-preserving language models, focusing on the transformer architecture with its Embedding, Encoder, and Decoder components, are increasingly trained in federated settings for enhanced privacy [LHZ⁺21, TWL⁺22]. Despite this progress, these models do not include tokenizers trained in federated settings. This limits achieving a truly end-to-end federated LM. Across the different frameworks proposed [LHZ⁺21] [CWW⁺23], tokenizers are initialized from an existing public tokenizer, even for domain-specific downstream tasks [YLW23], due to their parameterless structure and training process.

Previous works, such as the federated heavy hitters algorithm [ZKM⁺20], adapt distributed frequent sequence mining for word-level discovery but are primarily suited for word/subword WordPieces-based tokenizers. In [BSvD⁺22], tokenizers are learned through sampling

and auto-regressive text generative federated models, yet these approaches face limitations due to biases and high computational costs. However, these approaches are not applicable to the most widely used state-of-the-art subword/char/byte-level BPE-based tokenizers [SHB15, WCG20], defined as a subword tokenization technique for merging the most frequent pairs of characters or character sequences. Our focus is on BPE, which, unlike WordPieces, ensures no out-of-vocabulary (OOV) terms, and unlike SentencePiece [KR18], it does not require handling cross-boundary words. However, developing Federated Tokenizer faces several significant challenges: ensuring privacy preservation to prevent leakage of sensitive information during the training process, and achieving model convergence despite asynchronous updates from participants, which can lead to unstable convergence.

Contributions. In this chapter, we address these challenges by focusing our experiments on real-world financial data for efficiency, privacy, and robustness. Our contributions are as follows.

- We introduce a Federated Byte-level BPE Tokenizer algorithm to train a tokenizer across distributed datasets in real-world scenarios.
- We compare state-of-the-art generic and domain-specific pretrained tokenizers, highlighting their performances and trade-offs. Our results show that the federated tokenizer outperforms and competes with centralized pretrained tokenizers.
- We analyze the impact of federated tokenizer algorithm parameters, including the number of participants, partial data sharing, and privacy budget, on text compression performance. Our findings indicate that the privacy budget significantly affects performance, while partial data sharing acts as a regularization technique with minimal impact.

5.2 RELATED STUDIES

In this chapter, we are interested in federated learning, with a specific emphasis on the tokenization algorithms used in language models and the associated privacy concerns.

Tokenizers in Language Models.

Tokenization serves as a initial step in language models in both training and inference, transforms raw text into a model-understandable format. Techniques like WordPiece [WCG20], Byte Pair Encoding (BPE) [SHB15], SentencePiece [KR18], and Byte-level BPE [WCG20] are fundamental in language model pretraining. Recent studies have explored the impact and effectiveness of various tokenization techniques. Studies by [DRD19] and [TYŞO23a] reveal how tokenization strategies, from BPE merge counts to granularity levels in low-resource languages, influence model performance. Insights from [NTPV20] into BERT’s word-level tokenization highlight a preference for frequency over semantics, while [LBM23] discusses the selection of optimal tokenizers for multilingual models through extensive experimentation. Furthermore, the works by [RPV⁺21], [PLMTB24], and [BGBV22] explore the monolingual performance of multilingual models, the introduction of language biases, and the representation of OOV terms, respectively. However, these studies predominantly focus on the impact of tokenizers in multilingual settings, yet there remains a notable gap in examining their efficacy within domain-specific monolingual contexts.

Tokenizers in Federated Learning.

Studies focusing on language models within federated learning frequently employ word-level tokenization. While some research, such as [MRTZ17], constructs vocabularies from publicly available datasets, it is common to utilize standard pretrained tokenizers, subsequently fine-tuning the entire or partially language model for both generic [MRTZ17, AGM⁺22, KMS⁺21] and domain-specific downstream tasks [SR23, BRNM21]. Previous research on privately finding vocabulary items, such federated heavy hitters algorithm [ZKM⁺20], has been utilized for word-level discovery mainly from a single sequence per participant with a large privacy budget, rather than for training a tokenizer. With the rising attention on federated tokenizers, recent work by [BSvD⁺22] introduces a method for learning a tokenizer through sampling, leveraging auto-regressive text generative models. This method applies to both off-the-shelf and federated pre-trained models using a publicly trained tokenizer τ_{pub} , based on a public corpus. Then learns a new tokenizer by prompting the text generative model to generate new tokens. This approach, however, is limited by biases in the prediction

head, that reflect the word frequency used to train the public tokenizer in the pretraining corpus [KKYI23]. Additionally, the [BSvD⁺22] method requires training a language model in a federated learning setting, which is both expensive and unnecessary for training a tokenizer. Different from [ZKM⁺20, BSvD⁺22], our goal is to develop a bytes-level tokenizer specifically designed for a federated learning setting, by training on the vocabulary corpus instead of raw text itself. Researchers can potentially create different char-level tokenizers following our approach, if the tokenizer doesn't require to overlapping word boundaries, such SentencePiece.

5.3 PROBLEM STATEMENT

5.3.1 PRELIMINARIES

Tokenizer.

Tokenizers transform a string w into sequence of non-empty strings based on the vocabulary \mathcal{V} obtained through the training of a tokenizer algorithm on text corpus D . The function $\tau(w)$ represents the tokenization process, systematically producing all possible token sequences from w by repeatedly applying rules from \mathcal{V} until no further rules apply. Tokenization is defined formally in [BvdM23] as follows:

Definition 1. *Given an Σ as a finite set of symbols, the process of tokenization for a string $w \in \Sigma^*$ is the transformation of the string w into a sequence of tokens u_1, u_2, \dots, u_l , where each $u_i \in \Sigma^+$ for $1 \leq i \leq l$ given a collection of tokenization predefined rules \mathcal{V} that guide the segmentation of a string into tokens. The operation of recombining these tokens into the original string w is facilitated by a concatenation function π , such that $w = \pi(u_1, u_2, \dots, u_l)$.*

Federated Learning.

Federated learning trains a shared global model F using an aggregation protocol such as averaging involving N distributed participants (clients), each agreeing to train a local model f starting with the same initial configuration θ . Mathematically, let us assume that all N clients,

where the data reside, are available. Let D_i represent the data associated with client i , and n_i the number of samples available, with a total sample size is $\sum_{i=1}^N n_i$. Following [MRTZ17], this setup aims to solve an empirical risk minimization problem of the form :

$$\min_{\theta \in \mathbb{R}^d} F(\theta) = \sum_{i=1}^N \frac{n_i}{n} F_i(\theta) \quad \text{where} \quad F_i(\theta) = \frac{1}{n_i} \sum_{x \in D_i} f_i(\theta), \quad (5.1)$$

where d is the model parameters size to be learned.

Local Differential Privacy.

Local Differential Privacy (LDP) is a state-of-the-art privacy preservation technique that addresses the potential risk of an aggregation protocol at the server level to steal, expose, or leak clients' privacy over training rounds. In this setting, each client performs randomized perturbation on local data before sharing model parameters with the server, which then performs the aggregation protocol to obtain effective global model parameters. The same applies to statistics. LDP is defined formally in [KLN⁺11] as follows:

Definition 2. For a randomized algorithm M , its definition domain and range are $D(M)$ and $R(M)$, respectively. For any two records I and I' in $D(M)$, their same output of M is S , where $S \subseteq R(M)$. If the following inequality holds, then the randomized algorithm M satisfies (ε, δ) -LDP,

$$\Pr[M(I) = S] \leq e^\varepsilon \cdot \Pr[M(I') = S] + \delta. \quad (5.2)$$

where the parameter ε is called the privacy budget, refer to the privacy protection level of the client data can be adjusted through this parameter. And the parameter δ is called sensitivity of M , and represents the tolerance for the probability of ε deviating from its expected privacy guarantee, reflecting the maximum impact a single individual can have on the locally trained parameters or statistics. In practice, a larger ε means that the lower the privacy protection level M with a higher utility.

5.3.2 PROBLEM DEFINITION

In a federated setting comprising N participants denoted as $\{C_i\}_{i=1}^N$, each with a private local dataset D_i , our goal is to construct a federated tokenizer τ_{Fed} . This tokenizer should efficiently leverage all local datasets without compromising data privacy. The challenge lies in maximizing the similarity between τ_{Fed} and an hypothetical global tokenizer τ_{Global} , which would be constructed using a unified dataset $D = \bigcup_{i=1}^N D_i$. Formally, our goal is to:

$$\text{maximize } \text{sim}(\tau_{\text{Global}}, \tau_{\text{Fed}}) \quad (5.3)$$

Here, $\text{sim}(\cdot, \cdot)$ represents the similarity measure between τ_{Global} and τ_{Fed} . Assessing this similarity poses a unique challenge, as tokenizers with divergent training datasets and differing vocabulary sizes can yield equivalent outputs. To effectively assess this similarity, we propose utilizing metrics derived from the tokenization output of both tokenizers across the federation corpus. Specifically, we aim to:

$$\min_{\tau_{\text{Fed}}} \mathcal{F}(\tau_{\text{Fed}}) = \sum_{i=1}^N \frac{n_i}{n} \mathcal{F}_i(\tau_{\text{Fed}}) \quad \text{where} \quad \mathcal{F}_i(\tau_{\text{Fed}}) = \mathcal{H}(\tau_{\text{Fed}}(D_i)), \quad (5.4)$$

where $\mathcal{H}(\cdot)$ denotes a suitable metric computed over the tokenized output from τ_{Fed} applied to each local dataset D_i . This approach ensures the tokenizer efficacy in approximating τ_{Global} while preserving the privacy of the federated learning participants.

5.4 OUR APPROACH

In this section, we describe our methodology for creating a privacy-preserving tokenizer in a federated environment, aligned with data protection norms. Our approach starts with the local construction of vocabularies across participating entities, crucial for enhancing both privacy and effectiveness [DSR24], and beneficial for domain-specific NLP tasks [TYŞO23b]. We then implement a modified Byte Pair Encoding (BPE) algorithm within a federated learning framework [YLCT19], integrated with differential privacy [DR⁺14]. The aggregation

of these local vocabularies forms our federated BPE tokenizer, with further details in the following subsections.

5.4.1 PRIVACY-AWARE DATA PRE-PROCESSING

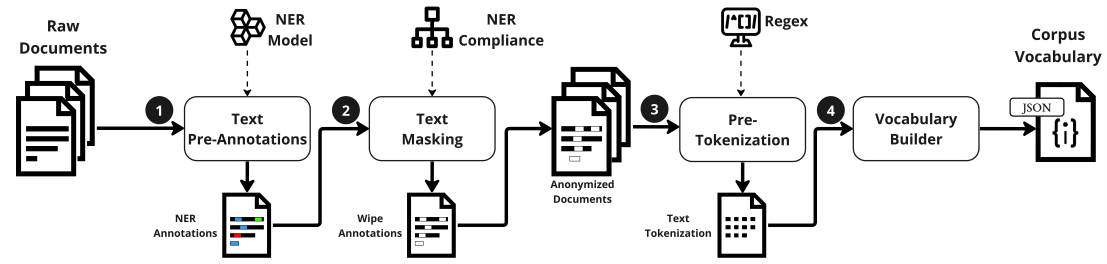


Figure 5.1: Automated Pipeline for Privacy-aware Vocabulary Corpus Builder.

Figure 1. shows a standardized pipeline for automated and privacy-aware vocabulary corpus builder, which can be enriched with different techniques depending on the use case. For financial domain, processing documents related to customer must to be aligned with GDPR in European Union and CCPA in the United State. The pipeline takes a financial document likely with personal information as input and returns a compliant corpus vocabulary as output.

Our designed pipeline is composed of four main steps, steps ①–④ illustrated in Figure 1, to ensure that the federated tokenizer operates without direct exposure to raw text data, thereby safeguarding the confidentiality and integrity of personal information throughout the tokenization process.

① **Text Pre-Annotations.** In this initial phase, a pretrained Named Entity Recognition (NER) pipeline is employed to systematically annotate the text within the input documents. The tagging operation classifies, or not, text segments in each document into specific categories based on contextual and the specific NER model applied. Our methodology rely on NERPII model [MMG23], specifically targets the identification of a NER entities related to PII, chosen specifically to enhance data privacy protection. Here are the categories targeted

due to their importance: Person Name, Email Address, Phone Number, Contract Number, Identification Number, and Zip Code / Address. Other distinct categories can be tagged, for further details on the NER model’s capabilities, we refer the reader to [LSHL22].

Below are the NER categories targeted due to their importance.

- Person Name: full/first/last names of individuals, could be an agent or the customer names.
- Email Address: identifies email addresses within the text.
- Phone Number: captures various formats of phone numbers.
- Contract Number: denotes alphanumeric sequences that represent more than mere words or numbers, including financial identifiers like IBANs and BICs.
- Identification Number: are unique identifiers such as social security or passport numbers.
- Zip Code / Address: recognizes patterns corresponding to zip codes and broader addressing schema.

In addition to outlined categories above, NER models are capable of identifying other distinct entity types depending on the use case, i.e. SetGNER for general NER tasks [HT22]. For further details on the NER model’s capabilities, we refer the reader to [LSHL22].

② **Text Masking.** In this step, we apply text anonymization to NER-identified segments within our entire collection of documents. Recent studies [PL23], [LLA⁺23], [LBR23] have explored various anonymization techniques—such as wiping, masking, or employing placeholders—and their impacts on language model performance alongside the balance between privacy and utility. In selecting the wiping technique for our federated tokenizer, we focused on its efficiency in removing specific NER-identified sensitive data, favoring a direct and less complex approach to minimize the potential for re-identification over other text anonymization techniques.

③ **Pre-Tokenization.** This step reply on the anonymized documents from steps ① and ② to automatically segment the text into granular segments, preparing it for the our tokenization algorithm by using a predefined regular expressions to break the input sequence of text into segments by greedily matching the following regular expression:

$$(?\\p\\{L\\}+|?\\p\\{N\\}+|?^\\s\\p\\{L\\}\\p\\{N\\}|\\r\\n|\\s+(?!\\S)|\\s+)$$

Inspired by recent studies on pre-tokenizers, our regex configuration is composed of digital-aware regex pattern, known as the Digit Tokenizer [CND⁺23, NJL21] in placing each digit separate segment, alongside with a punctuation-aware regex pattern, the Punct Tokenizer, based on promising results in [DSR24] for efficient tokenizer training.

④ **Vocabulary Builder.** Following pre-tokenization, this phase constructs the corpus vocabulary from the tokenized document outputs of step ③. Utilizing a counting function F , it counts each token's frequency across the anonymized corpus. The dataset $D = \{d_1, d_2, \dots, d_N\}$, encompasses all documents, with each d_i representing a sequence of tokens, denoted by $d_i = \{w_1^i, w_2^i, \dots, w_n^i\}$. Moreover, $W = \{w_1, w_2, \dots, w_m\}$ signifies the set of unique tokens within D , thus ensuring W is a subset of any $d \in D$, where $m \leq n$.

We can formally express the count function $F(D)$ as follows:

$$F(D) = \{(w_i, \sum_{d \in D} \text{count}(w_i, d)) \mid w_i \in W\}$$

This results in a set of tuples, represented by $\mathcal{D} = \{(w_i, f_i) \mid w_i \in W\}$, where each tuple pairs a unique token w_i with its frequency of occurrence within D , denoted by $f_i = F_D(w_i)$. In accordance with [WCG20] and [RWC⁺19], every token in W is converted into a byte sequence instead of Unicode characters, facilitating the training of byte-level tokenizers, where w_i will exclusively denote a sequence of bytes, underlining our focus on byte-level tokenization.

5.4.2 FEDERATED BPE TOKENIZER

Algorithm 4: Federated Byte-level BPE (1/2)

Require: clients $\mathcal{C} = \{C_i\}_{i=1}^N$, associated datasets $\{\mathcal{D}_i\}_{i=1}^n$

Parameters: k - Target vocab size, θ - threshold, ε and δ - privacy budget, p - Percentage of data from \mathcal{D}_i ,
K - Fraction of clients used in each round.

▷ Initial local tokenizer vocabulary

- 1: $V_{size} \leftarrow 256$
- 2: **for all** C_i in \mathcal{C} **do**
- 3: $C_i.vocab \leftarrow$ all possible byte values (0 to 255)
- 4: $C_i.merges \leftarrow \{\}$
- 5: **end for**
- ▷ Initial tokenizer training
- 6: **while** $V_{size} \leq k$ **do**
- 7: $\mathcal{C}' \leftarrow \text{RandomSampler}(\mathcal{C}, K)$ Single-Token Aggregation Phase
- 8: $L \leftarrow \{\text{ClientSendTuples}(C_i, \emptyset) \mid C_i \in \mathcal{C}'\}$
- 9: $U \leftarrow \text{ServerAggregateTuples}(L, \theta)$
- 10: $\mathcal{C}'' \leftarrow \text{RandomSampler}(\mathcal{C}, K)$ Token-Pair Aggregation Phase
- 11: $L \leftarrow \{\text{ClientSendTuples}(C_i, U) \mid C_i \in \mathcal{C}''\}$
- 12: $B \leftarrow \text{ServerAggregateTuples}(L, \theta)$
- ▷ Update Tokenizer and Local Vocab.
- 13: **if** $B = \emptyset$ **then**
- 14: **break** {No Vocabulary Left}
- 15: **end if**
- 16: $B_{max} \leftarrow B[0]$
- 17: $(t_L, t_R) \leftarrow B_{max}[0], B_{max}[1]$
- 18: **for all** $C_i \in \mathcal{C}$ **do**
- 19: $\text{ClientTupdateDataset}(C_i, t_L, t_R)$
- 20: **end for**
- 21: $V_{size} \leftarrow V_{size} + 1$
- 22: **end while**

We detail our Federated Byte-level BPE algorithm, an efficient, privacy-preserving tokenizer within a federated framework, in Algorithms 1 and 2. Following the BPE framework [SHB15, WCG20], participants C_i start with datasets \mathcal{D}_i (step 4 outcome). The algorithm initializes the local tokenizer vocabulary of byte values (0-255) and an empty set of merges (Algorithm

1, lines 1-5) then iteratively expands the vocabulary to size k through two main aggregation phases per iteration.

- **Single-Token Aggregation Phase** (Algorithm 1, lines 7-9): A subset of clients is randomly sampled, and they collectively send their single-token tuples to the server, which aggregates the tuples frequency of single-tokens and sends them back to a subset of clients.
- **Token-Pair Aggregation Phase** (Algorithm 1, lines 10-12): Following the single-token phase, a different subset of clients is sampled to receive the aggregated tuples from the server then generate token-pair tuples where one of token-pair must be in received single-tokens, and send back to the server to find the most frequent token-pair (Algorithm 2, lines 1-18).

If no vocabulary can be aggregated further (Algorithm 2, line 11) through all participants, indicating that B is empty (Algorithm 1, line 13), the process halts for that iteration and the target tokenizer vocabulary size is not reached. Otherwise, the most frequent tuple is identified (Algorithm 1, lines 16-17), and each client updates its local tokenizer vocabulary and dataset by merging the newly discovered byte pairs accordingly (Algorithm 1, lines 18-20). For a clearer understanding, here are the core procedures for client-server interactions in our system: **RandomSampler**(C, K) selects a subset K from set C , crucial for privacy and efficiency. **ClientSendTuples**(C_i, S) has clients generate tuples from data samples; if S is empty, only single tokens are processed and empty vocabularies return a special tuple. **ServerAggregateTuples**(C_i, θ) involves the server aggregating received tuples and applying a frequency threshold θ to maintain relevant data. **ClientUpdateDataset**(C_i, t_L, t_R) updates local vocabularies to reflect global changes, ensuring synchronization in the progress of federated learning.

Algorithm 5: Federated Byte-level BPE (2/2)

```

1: Procedure ClientSendTuples( $C_i, S$ ):
2:  $\mathcal{D}'_i \leftarrow \text{RandomSampler}(\mathcal{D}_i, p)$ 
3: if  $S = \emptyset$  then
4:    $\mathcal{U}'_i \leftarrow \text{GetUnigrams}(\mathcal{D}'_i)$ 
5:    $Singles \leftarrow \text{Sort}\{(t_L, f) \mid t_L \in \mathcal{U}'_i\}$ 
6:    $T \leftarrow \{(t_L, f + \mathcal{L}(0, \frac{\beta}{\epsilon})) \mid (t_L, f) \in Singles\}$ 
7: else
8:    $\mathcal{B}'_i \leftarrow \text{GetPairs}(\mathcal{D}'_i)$ 
9:    $Pairs \leftarrow \text{Sort}\{((t_L, t_R), f) \mid (t_L, t_R) \in \mathcal{B}'_i, t_L \in S\}$ 
10:  if  $Pairs = \emptyset$  then
11:    return ("NVL", 0) {No Vocabulary Left}
12:  else
13:     $T \leftarrow \{(pair, f + \mathcal{L}(0, \frac{\beta}{\epsilon})) \mid (pair, f) \in Pairs\}$ 
14:  end if
15: end if
16:  $T_{\max} \leftarrow T[0]$ 
17: return  $T_{\max}$ 
18: EndProcedure

19: Procedure ServerAggregateTuples( $L, \theta$ ):
20:  $T_{\text{pool}} \leftarrow \bigcup_{(key_i, f_i) \in L} \{(key_i, f_i)\}$ 
21:  $T_{\text{agg}} \leftarrow \{(key, \sum_{j \mid key_j = key} f_j)\}_{key \in T_{\text{pool}}}$ 
22:  $T_{\text{filtered}} \leftarrow \text{Sort}\{key \mid (key, f) \in T_{\text{agg}}, f > \theta\}$ 
23: return  $T_{\text{filtered}}$ 
24: EndProcedure

25: Procedure ClientUpdateDataset( $C_i, t_L, t_R$ ):
26:  $mergedToken \leftarrow t_L + t_R$ 
27: for each  $w$  in  $\mathcal{D}_i$  do
28:   if  $w$  contains  $t_L$  followed by  $t_R$  then
29:      $w \leftarrow \text{Replace } "t_L t_R" \text{ with } mergedToken \text{ in } w$ 
30:   end if
31: end for
32:  $C_i.vocab \leftarrow \text{Extend}(C_i.vocab, mergedToken)$ 
33:  $C_i.merges \leftarrow \text{Extend}(C_i.merges, (t_L, t_R))$ 
34: EndProcedure

```

After all iterative rounds, each client's tokenizer is generated for the target number of tokens k within $2k$ federated rounds, resulting in a tokenizer size of $256 + k$ unless the algorithm halts (Algorithm 1, line 14). We didn't study the communication overhead due to the limited bytes exchanged per round. However, a limitation of our work is the time complexity of $O(2kN \log N)$, which increases linearly with the number of rounds k , compared to the traditional BPE complexity of $O(N \log N)$ [ZMG⁺23].

5.4.3 PRIVACY PROTECTION

Here we highlight our efforts in protecting privacy by building a federated tokenizer through two privacy mechanisms.

First, we begin our algorithm with k as the target number of tokenizer vocabulary size, setting K as the number of clients to sample without replacement within a uniform distribution and θ as the minimal frequency for tokens. In every round, we derive two subsets from all the clients \mathcal{C} : with (a) uniformly sampled K clients without replacement $\mathcal{C}' \subset \mathcal{C}$, for single token selection, and (b) uniformly sampled K clients without replacement $\mathcal{C}'' \subset \mathcal{C}$, for pair token selection predicated on previously selected single tokens to extend the already discovered merges (pair tokens), with respect to the thresholding operation θ . This approach ensure not only k -anonymity properties but also reinforces differential privacy, as demonstrated in [ZKM⁺20, Theorem 1], where $k = \theta$.

Given our dual-faceted process, we define \mathcal{M}_1 as a global privacy mechanism, inherently differentially private, as substantiated by [ZKM⁺20, Den80]. In instances where a client \mathcal{C}_i is randomly chosen in both \mathcal{C}' and \mathcal{C}'' for the discovery of token pairs, only those pairs that exceed the frequency threshold θ are evaluated. Thus, \mathcal{M}_1 improves a form of privacy by locally excluding low-frequent tokens, potentially identifiable.

Finally, the local private token frequency is carried out using a different mechanism \mathcal{M}_2 relying on Local Differential Privacy, denoted by (ϵ, δ) -LDP. First, we define $p \in [0, 1]$ as the subset ratio for sampling local client dataset, where we subsample $\mathcal{D}'_i \subset \mathcal{D}_i$, following a

uniform distribution without replacement, then we add a controlled random noise sampled from Laplace distribution to the local frequency distribution of single and pair tokens, $F_{D'_i}$. As a result, the \mathcal{M}_2 mechanism enhance privacy and retains utility by using both subsampling [BBG20] and for noise addition [KLN⁺11].

Following our second mechanism, let w be a single or pair token in \mathcal{D}_i and due to the randomness inherent in the uniform subsampling process used in \mathcal{M}_2 , the expected frequency can be expressed as;

$$E[F_{D'_i}(w)] = p \times F_{D_i}(w);$$

where $E[F_{D'_i}(w)]$ is the expected frequency of token w in the D_i . Due to the subsampling perturbation introduced, D'_i might not perfectly reflect the distribution of D_i , this process can be likened to the effects of BPE-dropout [PEV19]. By integrating the Laplace noise and subsampling, we ensures a deviation in the original distribution that will effectively mask these variations with a privacy budget ε .

$$|F_{D'_i}(w) - F_{D'_i, \text{noisy}}(w)| \leq \varepsilon$$

Although both mechanisms increase the level of privacy separately, their sequential combination, as described in the composition theorem [DR⁺14], forms a unified mixture mechanism \mathcal{M} . This integration involves employing \mathcal{M}_1 to protect against identifying rare token patterns and \mathcal{M}_2 to add controlled random noise to the local frequency distributions of tokens. The resultant mechanism \mathcal{M} achieves $\varepsilon' = \ln(1 + p(e^\varepsilon - 1))$ and $\delta' = p\delta$, according to [BBG18, Theorem 9], while not relying solely on k anonymity for privacy [ZKM⁺20].

5.5 DESIGN OF EXPERIMENTS

In this section, we describe the datasets and metrics used and the experiment set up to assess and evaluate the efficacy of the proposed FedByteLevelBPE algorithm.

5.5.1 REAL DATASET

We utilize the CFPB* open dataset, initially for risk monitoring in financial consumer services, now repurposed for federated learning in NLP. Unlike previous studies [LDCH22], we distributed the data set between 30 US financial institutions based on the CompanyName, encompassing 680,086 complaints from January 2016 to May 2023, totaling about 90 million tokens. This partitioning forms a natural cross-silo setup, mirroring real-world data variations and is processed uniformly as described in section 5.4.1.

5.5.2 EVALUATION METRICS

Tokenizer performance evaluates the relationship between input text and output lengths to assess how effectively the tokenizer encodes and compresses data. We focus on two metrics: tokenizer’s fertility (ψ), which measures the average subtokens per token, reflecting granularity ($\psi \geq 1$ indicates optimal dataset tailoring), given by:

$$\psi(\tau) = \frac{1}{|D|} \sum_{s \in D} \frac{|\tau(s)|}{|s|} \quad (5.5)$$

and the proportion of continued words (Π), which assesses the tokenizer’s tendency to split words, calculated by:

$$\Pi(\tau) = 1 - \frac{|D|}{\sum_{s \in D} |\tau(s)|} \quad (5.6)$$

These metrics, ψ and Π , scale from individual tokens to datasets by considering D as either concatenated sentences for vocabulary-level or as document collections for broader evaluations.

*The Consumer Financial Protection Bureau (CFPB) database: [link](#)

Table 5.1: Hyperparameter Tuning Space for the FedByteLevelBPE Algorithm

Hyperparameter	Definition	Search Interval
θ	Minimal frequency per tokens	$\{8, 15, 22, 30\}$
ε	Privacy budget for (ε, δ) -LDP	$\{10^{-3}, 5 \cdot 10^{-3}, 5 \cdot 10^{-2}, 10^{-1}, 5 \cdot 10^{-1}, 1, 2\}$
p	Percentage of data from D_i	$\{0.7, 0.8, 0.9, 0.95, 1.0\}$
K	Fraction of clients in each round	$\{0.25, 0.5, 0.75, 1.0\}$

5.5.3 APPROACH FOR COMPARISON

We assess our algorithm against recent tokenizer models by classifying tokenizers into three types: **Public Tokenizer** τ_{pub} , using a general corpus; **Local Tokenizer** τ_{loc} , trained on data from a single institution; and **Federated Tokenizer** τ_{fed} , developed in a federated learning context. Each category utilized an identical setup with a vocabulary size of 50257. Specifically, 30 local tokenizers were trained for individual financial institutions, and a federated tokenizer aggregated insights from all datasets, with performance metrics averaged across federated datasets. The details of the hyperparameters of the FedByteLevelBPE algorithm are detailed in Table 5.1, where the tokenizer vocabulary is fixed at 50257, δ is set to 1 according to [AFG16], and a total of 560 hyperparameter configurations are evaluated across the 30 client nodes. These evaluations were conducted using the flwr framework on a server with a 32 core CPU and 128 GB RAM running Ubuntu 22.04.

5.6 EXPERIMENTS AND RESULTS

5.6.1 HYPERPARAMETER TUNING RESULTS

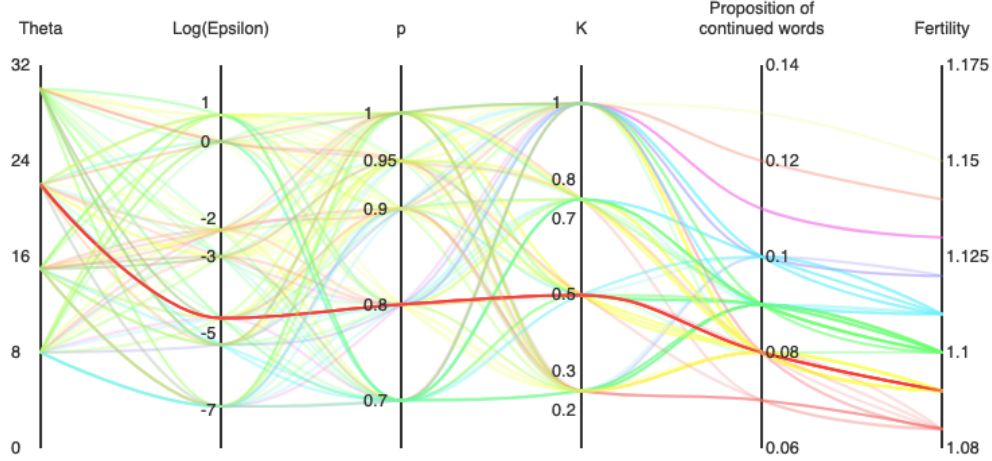


Figure 5.2: Parallel Coordinates Analysis of 560 Hyperparameter Combinations Affecting Federated Tokenizer Performance: Fertility and Proportion of Continued Words. Optimal parameters are highlighted with a red line.

In this section, we conducted a comprehensive hyperparameter tuning for FedByteBPE, exploring the sensitivity of the algorithm to each parameter through a grid search detailed in Table 5.1. Figure 5.2 shows the impact of the parameters on the tokenizer’s performance at the document level, particularly metrics Π and ψ .

In our analysis of the privacy parameter ε , we explored the lower and upper bounds of the privacy impact on the tokenizer’s performance relative to our ground truth, the hypothetical centralized tokenizer, with ψ of 1.07 and Π of 0.06 as upper bound benchmark for our comparison.

- **High Privacy Budget ($\varepsilon \geq 10^{-2}$):** When ε is set higher, less noise is introduced, resulting in better performance metrics that closely align with those of the hypothetical centralized tokenizer. Specifically, the fertility ψ exhibits a minimal decrease ranging

from -2.8% to -0.93%, compared to the centralized tokenizer. This indicates high utility with minor privacy trade-offs. Similarly, the proportion of continued words Π also shows slight decrease, ranging from -1.67% to -1.33%, suggesting better preservation of linguistic structure under lower noise conditions. However, this setting carries a potential risk of exposing sensitive information due to minimal data perturbation.

- **Low Privacy Budget ($\varepsilon \leq 5 \cdot 10^{-4}$):** A lower ε adds more noise, significantly enhancing data privacy but reducing utility. This leads to greater deviation from the centralized tokenizer's performance, with the proportion of continued words Π and fertility ψ showing a decrease, ranging from -3.0% to -2.0% and -7.48% to -3.74% respectively. Under these conditions, the tokenizer tends to generate more subtokens per input token at the document level, reflecting decreased efficiency and a stronger emphasis on privacy.

In a high privacy budget setting, our analysis revealed that sub-sampling of data, parameterized by p , significantly enhances tokenizer construction by introducing diversity or regularization, while the number of clients (K) has a marginal impact, suggesting redundancy in vocabulary does not significantly alter performance beyond a certain participant threshold. Furthermore, increasing θ narrows the vocabulary, enhancing tokenizer efficiency. These optimizations led to a refined performance close to our ground truth emphasizing the robustness of θ and p in achieving optimal tokenizer functionality in federated settings without the need for excessively large numbers of participants per round.

To this end, identifying a universal set of parameters that optimally balance privacy and utility proved challenging. However, the proposed algorithm demonstrated considerable robustness in terms of participant numbers and the volume of shared data, within the constraints of a given privacy budget. Consequently, we have empirically selected the best hyperparameters (marked by the red line in Figure 5.2), where $\theta = 24$, $\varepsilon = 0.01$, $p = 0.8$, and $K = 0.5$ are used to build the the Federated Tokenizer τ_{fed} . The τ_{fed} tokenizer achieve ψ of 1.09 and Π of 0.08, representing decreases of -1.86% and -1.67%, respectively, compared to the hypothetical centralized tokenizer.

5.6.2 PERFORMANCE COMPARISON ACROSS TOKENIZERS

This section provides a comparative analysis of the performance of various tokenizer models. Table 5.1 show state of the art generic and domain specific pretrained tokenizer, where the reported performance is the average cross 30 distributed datasets, same for local, centralized and federated tokenizers. We focus on two primary metrics in section 5.5.2 at both the vocabulary and document levels.

At the vocabulary level, public tokenizers pretrained on generic corpora exhibit higher fertility ψ rates and proportion of continued words Π . On the other hand, the performance of domain-specific tokenizers show more tailored performance. However, models like FinGPT and FinMegatronGPT that supplement their training data with public corpora due to limited open-source financial datasets shows a decrease in the Π by approximately 1.9% compared to BERT, reflecting a modest enhancement in maintaining more original word forms. Still GPT_{3.5}-Turbo stands out among generic models with a competitive performance that approaches domain-specific models due to its large-scale training base and the extremely high tokenizer size. Seeking more efficiency and smaller models, FinBERT, FinancialBERT and FLANG-BERT shows a best fertility ψ with a significantly lower proportion of continued words Π . Given that the vocabulary level metrics treat all unique corpus words with equal importance, we augmented our analysis with document level performance metrics, here fertility is weighted by the frequency of each word in the corpus. Similarly, the proportion of continued words is adjusted by the extent to which a document's length changes.

Model Name	Size	Vocabulary Level		Document Level	
		ψ	Π	ψ	Π
Off-The-Shelf Domain-General Tokenizers					
$\tau_{\text{BERT}}[\text{KT}^{19}]$	30,522	2.18 \pm 0.21	0.54 \pm 0.04	1.13 \pm 0.03	0.10 \pm 0.02
$\tau_{\text{BART}}[\text{LLG}^{+19}]$	50,265	2.14 \pm 0.12	0.53 \pm 0.02	1.21 \pm 0.02	0.17 \pm 0.01
$\tau_{\text{GPT}_2}[\text{RWC}^{+19}]$	50,257	2.14 \pm 0.12	0.53 \pm 0.02	1.21 \pm 0.02	0.17 \pm 0.01
$\tau_{\text{GPT}_{3.5}\text{-Turbo}}^{\dagger}$	100,261	1.97 \pm 0.11	0.49 \pm 0.03	1.16 \pm 0.02	0.13 \pm 0.01
$\tau_{\text{T}_5}[\text{RSR}^{+20}]$	32,100	2.01 \pm 0.18	0.50 \pm 0.04	1.11 \pm 0.02	0.10 \pm 0.01
$\tau_{\text{Llama}_2}[\text{TLI}^{+23}]$	32,000	2.13 \pm 0.15	0.53 \pm 0.03	1.15 \pm 0.02	0.12 \pm 0.01
Average		2.09 \pm 0.15	0.52 \pm 0.03	1.16 \pm 0.02	0.13 \pm 0.01

Model Name	Size	Vocabulary Level		Document Level	
		ψ	Π	ψ	Π
Off-The-Shelf Domain-Specific Tokenizers					
$\tau_{\text{FinGPT}}[\text{YLW}^{23}]$	32,000	2.13 \pm 0.15	0.53 \pm 0.03	1.15 \pm 0.02	0.12 \pm 0.01
$\tau_{\text{FinBERT}}[\text{LHH}^{+21}]$	30,873	1.91 \pm 0.13	<u>0.30\pm0.05</u>	1.08 \pm 0.01	0.07 \pm 0.01
$\tau_{\text{SecBERT}}^{\ddagger}$	30,000	1.95 \pm 0.12	0.36 \pm 0.05	<u>1.07\pm0.01</u>	0.08 \pm 0.01
$\tau_{\text{FLANG-BERT}}[\text{SCE}^{+22}]$	30,522	1.90 \pm 0.14	0.37 \pm 0.05	1.07 \pm 0.01	0.07 \pm 0.01
$\tau_{\text{FinancialBERT}}[\text{Haz}^{22}]$	30,873	<u>1.88\pm0.09</u>	0.44 \pm 0.03	1.11 \pm 0.02	0.09 \pm 0.01
$\tau_{\text{FinanceDeBERTa}}[\text{WLQ}^{+22}]$	128,000	1.91 \pm 0.13	0.32 \pm 0.06	1.08 \pm 0.01	<u>0.06\pm0.01</u>
$\tau_{\text{FinMegatronGPT}}[\text{Wu}^{21}]$	50,257	2.14 \pm 0.12	0.53 \pm 0.02	1.21 \pm 0.02	0.17 \pm 0.01
$\tau_{\text{FinMegatronBERT}}[\text{Wu}^{21}]$	30,522	2.11 \pm 0.14	0.47 \pm 0.05	1.16 \pm 0.01	0.15 \pm 0.01
$\tau_{\text{FinanceDistilGPT}_2}^{\S}$	50,257	2.14 \pm 0.12	0.53 \pm 0.02	1.21 \pm 0.02	0.17 \pm 0.01
Average		2.00 \pm 0.13	0.43 \pm 0.04	1.13 \pm 0.01	0.11 \pm 0.01

Model Name	Size	Vocabulary Level		Document Level	
		ψ	Π	ψ	Π
Local Domain-Specific Tokenizers					
τ_{Equifax}	50,257	<u>2.08\pm0.11</u>	<u>0.52\pm0.03</u>	<u>1.11\pm0.02</u>	<u>0.10\pm0.02</u>
τ_{Experian}	50,257	<u>2.08\pm0.11</u>	<u>0.52\pm0.03</u>	<u>1.11\pm0.02</u>	<u>0.10\pm0.02</u>
$\tau_{\text{TransUnion}}$	50,257	2.10 \pm 0.11	0.52 \pm 0.03	1.12 \pm 0.02	0.11 \pm 0.02
$\tau_{\text{BankOfAmerica}}$	50,257	2.13 \pm 0.12	0.53 \pm 0.03	1.14 \pm 0.01	0.12 \pm 0.01
$\tau_{\text{JPMorganChase}}$	50,257	2.13 \pm 0.12	0.53 \pm 0.03	1.14 \pm 0.01	0.12 \pm 0.01
τ_{Citibank}	50,257	2.15 \pm 0.12	0.53 \pm 0.03	1.14 \pm 0.02	0.12 \pm 0.01
$\tau_{\text{CapitalOne}}$	49,724	2.18 \pm 0.12	0.54 \pm 0.03	1.14 \pm 0.02	0.12 \pm 0.01
$\tau_{\text{WellsFargo}}$	50,257	2.13 \pm 0.12	0.53 \pm 0.03	1.14 \pm 0.01	0.12 \pm 0.01
τ_{Navient}	38,790	2.32 \pm 0.13	0.57 \pm 0.02	1.22 \pm 0.02	0.17 \pm 0.01
$\tau_{\text{Synchrony}}$	40,141	2.29 \pm 0.13	0.56 \pm 0.02	1.19 \pm 0.02	0.15 \pm 0.02
τ_{Amex}	35,419	2.36 \pm 0.12	0.57 \pm 0.02	1.23 \pm 0.02	0.18 \pm 0.02
$\tau_{\text{U.S.Bank}}$	37,226	2.34 \pm 0.12	0.57 \pm 0.02	1.22 \pm 0.02	0.17 \pm 0.01
$\tau_{\text{PortfolioRecovery}}$	24,654	2.54 \pm 0.12	0.60 \pm 0.02	1.30 \pm 0.04	0.23 \pm 0.02
τ_{PayPal}	31,137	2.45 \pm 0.13	0.59 \pm 0.02	1.27 \pm 0.03	0.21 \pm 0.02
$\tau_{\text{BreadFinancial}}$	28,016	2.48 \pm 0.12	0.60 \pm 0.02	1.27 \pm 0.03	0.21 \pm 0.02
τ_{Discover}	32,855	2.39 \pm 0.12	0.58 \pm 0.02	1.25 \pm 0.02	0.19 \pm 0.01
$\tau_{\text{Nationstar}}$	36,530	2.35 \pm 0.12	0.57 \pm 0.02	1.23 \pm 0.02	0.19 \pm 0.01
τ_{AES}	28,364	2.47 \pm 0.12	0.59 \pm 0.02	1.27 \pm 0.02	0.21 \pm 0.01
τ_{Ocwen}	36,203	2.36 \pm 0.12	0.58 \pm 0.02	1.24 \pm 0.02	0.19 \pm 0.01
$\tau_{\text{EncoreCapital}}$	24,206	2.55 \pm 0.12	0.61 \pm 0.02	1.31 \pm 0.04	0.23 \pm 0.03
τ_{TDBank}	29,585	2.45 \pm 0.12	0.59 \pm 0.02	1.27 \pm 0.02	0.21 \pm 0.01
τ_{PNC}	28,940	2.47 \pm 0.12	0.59 \pm 0.02	1.28 \pm 0.02	0.21 \pm 0.01
τ_{Barclays}	26,127	2.51 \pm 0.12	0.60 \pm 0.02	1.29 \pm 0.03	0.22 \pm 0.02
$\tau_{\text{Santander}}$	25,305	2.54 \pm 0.12	0.61 \pm 0.02	1.31 \pm 0.03	0.23 \pm 0.02
τ_{Ally}	24,917	2.55 \pm 0.12	0.61 \pm 0.02	1.33 \pm 0.02	0.24 \pm 0.01
$\tau_{\text{ResurgentCapital}}$	20,066	2.66 \pm 0.12	0.62 \pm 0.02	1.36 \pm 0.04	0.26 \pm 0.02
τ_{USAA}	27,716	2.49 \pm 0.12	0.60 \pm 0.02	1.30 \pm 0.02	0.23 \pm 0.01
τ_{ERC}	18,165	2.70 \pm 0.12	0.63 \pm 0.02	1.38 \pm 0.04	0.27 \pm 0.02
$\tau_{\text{NavyFederal}}$	26,268	2.51 \pm 0.12	0.60 \pm 0.02	1.31 \pm 0.02	0.23 \pm 0.01
τ_{Coinbase}	19,605	2.69 \pm 0.12	0.63 \pm 0.02	1.42 \pm 0.03	0.29 \pm 0.02
Average		2.38 \pm 0.12	0.58 \pm 0.02	1.24 \pm 0.02	0.19 \pm 0.01

Model Name	Size	Vocabulary Level		Document Level	
		ψ	Π	ψ	Π
Centralized/Federated Domain-Specific Tokenizers					
$\tau_{\text{Centralized}}$	50,257	<u>2.03\pm0.12</u>	<u>0.51\pm0.03</u>	1.07 \pm 0.01	<u>0.06\pm0.01</u>
$\tau_{\text{Federated}}$	50,257	2.04 \pm 0.11	<u>0.51\pm0.03</u>	1.09 \pm 0.01	0.08 \pm 0.01

This section provides a comparative analysis of the performance of various tokenizer models. Table 5.1 show state of the art generic and domain specific pretrained tokenizer, where the reported performance is the average cross 30 distributed datasets, same for local, centralized and federated tokenizers. We focus on two primary metrics in Section 5.5.2 at both the vocabulary and document levels.

At the vocabulary level, public tokenizers pretrained on generic corpora exhibit higher fertility rates ψ and the proportion of continued words Π . On the other hand, the performance of domain-specific tokenizers shows a more tailored performance. However, models like FinGPT and FinMegatronGPT that supplement their training data with public corpora due to limited open-source financial datasets shows a decrease in the Π by approximately 1.9% compared to BERT, reflecting a modest enhancement in maintaining more original word forms. Still GPT3.5-Turbo stands out among generic models with a competitive performance that approaches domain-specific models due to its large-scale training base and the extremely high tokenizer size. Seeking more efficiency and smaller models, FinBERT, FinancialBERT and FLANG-BERT shows a best fertility ψ with a significantly lower proportion of continued words Π . Given that the vocabulary level metrics treat all unique corpus words with equal importance, we augmented our analysis with document level performance metrics, here fertility is weighted by the frequency of each word in the corpus. Similarly, the proportion of continued words is adjusted by the extent to which a document's length changes.

At the document level, the performance differences between generic and domain-specific tokenizers are even more pronounced. Excluding domain-specific GPT-based models and BPE-based tokenizers, generic models do not outperform their domain-specific counterparts. The T5, as the best generic model, shows a ψ of 1.11 and a Π of 0.10, while FLANG-BERT, the

best domain-specific model, outperforms it with a ψ of 1.07 and a Π of 0.07. This represents a 3.6% improvement in fertility and a 30% reduction in word splits by FLANG-BERT compared to T5, highlighting the substantial benefits of domain-specific tokenizers in reducing unnecessary fragmentation and maintaining semantic integrity in specialized fields.

Local trained tokenizers developed for specific financial institutions, such as Bank of America, Citibank, and Wells Fargo, perform slightly better than generic models. However, they are less effective compared to domain-specific models like FLANG-BERT. In a federated learning environment, the Federated tokenizer $\tau_{Federated}$ exhibits competitive capabilities, with a ψ of 1.09 and a Π of 0.08, closely approaching the performance of the hypothetical centralized tokenizer $\tau_{Centralized}$ with ψ of 1.07 and Π of 0.06. This shows that Federated tokenizer not only outperforms all BPE-based and most WordPiece-based tokenizers but also provides a robust privacy-preserving solution that nearly matches the top-performing domain-specific model FLANG-BERT as centralized model in maintaining linguistic integrity in tokenization.

5.6.3 BI-VARIATE ANALYSIS OF TOKENIZER CHOICES

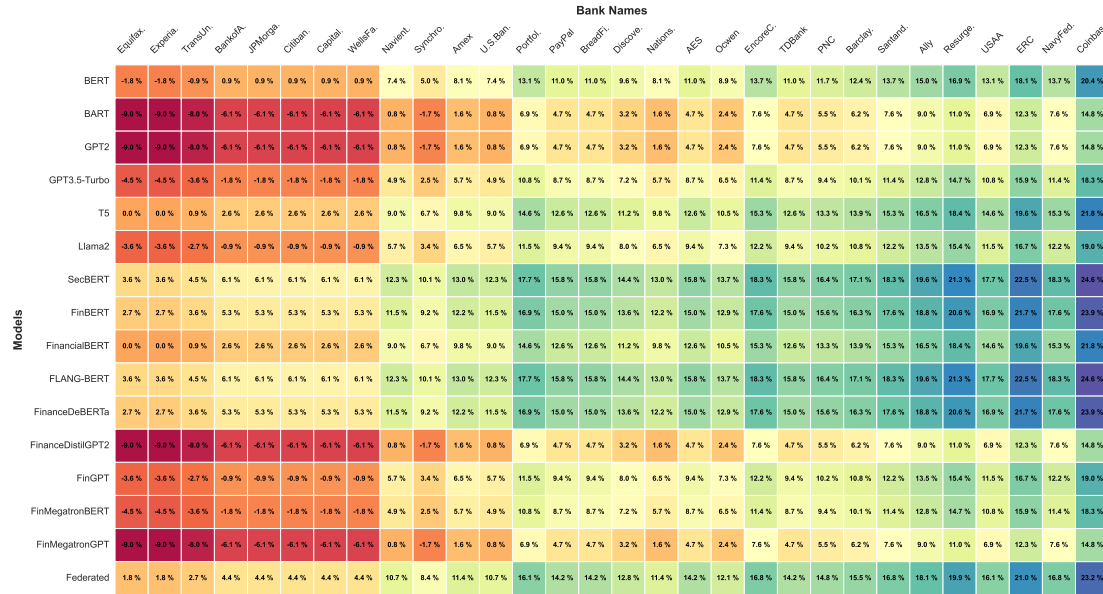


Figure 5.3: Parity Plot.

In our evaluation of the federated tokenizer, the model demonstrated strong performance across a range of financial institutions, consistently outperforming a significant number of pre-trained tokenizers that were trained in a centralized setting. Specifically, the federated tokenizer outperformed 75% of the pre-trained tokenizers for most of the participating institutions, as indicated by the results for Equifax, Experian, TransUnion, Bank of America, JPMorgan Chase, Citibank, Capital One, Wells Fargo, and others. This indicates that the federated model was superior to approximately 11 out of 15 pre-trained models in these cases, for both key performance metrics: fertility (ψ) and the proportion of continued words (Π).

For certain institutions like Navient, U.S. Bank, and Barclays, the federated tokenizer outperformed 87.5% of the pre-trained tokenizers, equating to surpassing about 13 out of 15 models. These results highlight the federated model's ability to generalize well across both generic and domain-specific tokenizers, irrespective of the underlying architecture (whether BERT-based, GPT-based, or other). Consistent performance across different institutions

and model families underscores the strength of the federated learning approach in creating domain-specific tokenizers that are not only competitive but often superior to centrally trained tokenizers.

The federated tokenizer’s ability to integrate knowledge from distributed datasets, while preserving privacy, positions it as a powerful tool in specialized domains like finance, where data sensitivity and domain-specific language are critical. This success across different model architectures further suggests that federated learning could be an effective strategy for developing robust tokenizers in other domains requiring tailored language processing capabilities.

However, it is also noteworthy that some domain-specific tokenizers, such as FLANG-BERT, FinBERT, FinanceDeBERTa, and SecBERT, outperformed the federated model in certain cases. These models often leveraged a centralized corpus that included data from all participating institutions, giving them a broader perspective and an advantage in capturing domain-specific language nuances. Moreover, the use of word-based tokenization in many BERT-based models, while effective, can sometimes lead to out-of-vocabulary (OOV) issues, a challenge partially mitigated by the extensive training data available in centralized settings.

Overall, the federated model’s performance, outperforming between 75% to 87.5% of the pre-trained tokenizers, highlights its efficacy and potential as a privacy-preserving solution in specialized domains. However, it also suggests that further refinement, potentially incorporating techniques from domain-specific models, could enhance its performance even further.

5.7 CONCLUSION

In this chapter, we presented the Federated Byte-Level BPE Tokenizer (FedByteBPE), a privacy-preserving language model tokenizer leveraging federated learning. This approach allows entities to train tokenizers locally, with centralized vocabulary aggregation ensuring robust, domain-specific tokenization. Our empirical analysis, conducted on real-world financial datasets, demonstrated that FedByteBPE consistently outperforms off-the-shelf and locally trained to-

kenizers in vocabulary coverage and document-level performance, highlighting its effectiveness for distributed language processing in the financial sector.

6

Conclusion and Future Work

CHAPTER CONTENTS

6.1	Conclusion	138
6.2	Future work	139

6.1 CONCLUSION

In this dissertation, we presented multiple methodologies to address some of the critical challenges inherent in collaborative learning environment for the financial sector. The first contribution of this work is the development of G-HIN2Vec [DSHS23a], a graph-level embedding method designed specifically for heterogeneous information networks for cardholder representation. By leveraging an unsupervised learning framework trained with a double-triplet loss function, G-HIN2Vec effectively captures the complex relationships embedded within financial networks. Through conducted experimentation, we found that G-HIN2Vec outperformed existing methods in cardholder spending behavior representation by encoding socio-demographic features, achieving a 2.45% improvement in gender classification accuracy and a 7.19% increase in income prediction R-squared (R^2). Additionally, the model reduced the mean absolute error (MAE) in age prediction by 6.55%, demonstrating its robustness in handling real-world financial data. The experiments show that DiffPool [YYM⁺18] revealed a competitive performance and simplicity of homogeneous graph representations compared to G-HIN2Vec. This realization led to a strategic shift in focus toward homogeneous graph representation learning, which proved more adept at addressing the dynamic aspects of cardholder transactions.

We extend cardholder graph modeling introduced as anonymization-based privacy mechanism to a novel privacy-preserving behavioral anomaly detection framework. This framework integrates Graph Neural Networks (GNNs) with domain-specific graph-based negative sampling techniques incorporated to generate contextually relevant labels within dynamic graphs for training anomaly detection model across the federation. Our negative sampling strategy not only enables anomaly detection training but also minimizes the alignment efforts required from federation participants to train anomaly and fraud detection models. Our results showed that this approach achieved F1-scores of 0.91 ± 0.02 on synthetic data and 0.87 ± 0.04 on real-world data, outperforming traditional methods by a significant margin. In addition, we implemented a model stacking strategy by integrating graph-based federated anomaly scores into a centralized and federated fraud detection model, which enhanced both the models. Reflecting on this, it was rewarding to see how combining centralized and

federated approaches could produce a more resilient fraud detection system, offering privacy protection without compromising on performance.

Privacy, particularly the protection of personally identifiable information (PII), has been a central concern throughout this dissertation. To address this, we introduced novel privacy mechanisms designed for PII within the federated learning framework, as an ego-centric static and dynamic graphs, effectively removing the need for unique identifiers in the FL model. However, recognizing the persistent privacy risks posed by sensitive attributes like location and timestamps, we extended Personalized Local Differential Privacy (PLDP) to include a personalized noising mechanism for temporal and spatial data attributes. This ensures that even if a data breach occurs, the raw data remains protected while still enabling the FL model to learn relevant patterns.

Lastly, this dissertation presents the Federated Byte-Level BPE Tokenizer, an innovative approach for privacy-preserving language model tokenization across distributed datasets. This tokenizer is designed to function within a privacy-aware vocabulary corpus, preserving personally identifiable information. Our methodology enables the training of a federated tokenizer that surpasses locally trained and existing pretrained language models in terms of vocabulary coverage and processing efficiency. By acting as a domain-specific tokenizer, it maintains robust data privacy while delivering superior performance. The success of Fed-ByteBPE underscores the significant potential of federated learning in natural language processing, providing a secure and efficient solution for institutions requiring privacy-preserving language model training.

6.2 FUTURE WORK

Building on the methodologies and findings presented in the thesis, there are several promising directions for future research to further enhance the application and effectiveness of FL in the financial sector.

Federated Tokenizers and Language Models: Future research will explore the development of FL-based tokenizers, extending their capabilities to other tokenizer types such as

WordPiece and SentencePiece. These advancements target is to improve privacy in NLP downstream tasks within federated environments. A particularly promising direction is the concept, we call Minimal Tokenizer Adaptation, which involves adapting federated tokenizers to large language models without requiring the re-training, using transfer learning paradigms, this approach target to minimize computational overhead while preserving model effectiveness.

Enhanced Privacy Mechanisms: Privacy remains a cornerstone of FL, especially in sensitive financial applications. Future work will focus on integrating privacy-preserving techniques at every stage of the machine learning workflow, for example in our work we focus on feature engineering, to develop privacy-aware end-to-end FL frameworks.

This research will involve a comprehensive exploration of privacy-enhancing technologies, such as cryptographic methods, hardware-based solutions, anonymization techniques to reduce the risk of re-identification, and noise-based approaches, such as differential privacy. By embedding privacy mechanisms throughout the machine learning workflow, FL systems can ensure robust data protection while maintaining performance and scalability.

Extended Federated Learning Applications of Inductive Graph Learning: Future research could target inductive graph learning within FL to enhance model utility and adaptability in evolving datasets. This includes applications in areas such as credit scoring, community detection, and transaction monitoring. While our earlier work focused on transductive learning on static graphs, our recent efforts have shifted toward developing inductive models capable of generalizing to unseen data on dynamic graphs.

Future work will focus on integrating inductive graph learning into FL environments, leveraging self-supervised learning techniques such as negative sampling to increase adaptability of federated solutions and enable efficient training. Addressing key challenges, such as balancing trade-offs between model accuracy, computational cost, and communication overhead, will be essential to developing scalable and effective FL frameworks.

Appendices



Language Model Tokenizer Performance Improvement Across Models

Table A.1: Comparative Analysis of Tokenizer Fertility Improvement (φ) at the Document Level Across Local, Pre-trained, and Federated Approaches

	BERT	BART	GPT ₂	GPT _{3.5} .Turbo	T ₅	Llama2	SecBERT	FinBERT	FinancialBERT	FLANG.BERT	FinanceDeBERTa	FinanceDistilGPT ₂	FinGPT	FinMegatronBERT	FinMegatronGPT	Centralized	Federated
Equifax	-1.8%↓	-9%↓	-9%↓	-4.5%↓	0%	-3.6%↓	3.6%↑	2.7%↑	0%	3.6%↑	2.7%↑	-9%↓	-3.6%↓	-4.5%↓	-9%↓	3.6%↑	1.8%↑
Experian	-1.8%↓	-9%↓	-9%↓	-4.5%↓	0%	-3.6%↓	3.6%↑	2.7%↑	0%	3.6%↑	2.7%↑	-9%↓	-3.6%↓	-4.5%↓	-9%↓	3.6%↑	1.8%↑
TransUnion	-0.9%↓	-8%↓	-8%↓	-3.6%↓	0.9%↑	-2.7%↓	4.5%↑	3.6%↑	0.9%↑	4.5%↑	3.6%↑	-8%↓	-2.7%↓	-3.6%↓	-8%↓	4.5%↑	2.7%↑
Bank of America	0.9%↑	-6.1%↓	-6.1%↓	-1.8%↓	2.6%↑	-0.9%↓	6.1%↑	5.3%↑	2.6%↑	6.1%↑	5.3%↑	-6.1%↓	-0.9%↓	-1.8%↓	-6.1%↓	6.1%↑	4.4%↑
JPMorgan Chase	0.9%↑	-6.1%↓	-6.1%↓	-1.8%↓	2.6%↑	-0.9%↓	6.1%↑	5.3%↑	2.6%↑	6.1%↑	5.3%↑	-6.1%↓	-0.9%↓	-1.8%↓	-6.1%↓	6.1%↑	4.4%↑
Citibank	0.9%↑	-6.1%↓	-6.1%↓	-1.8%↓	2.6%↑	-0.9%↓	6.1%↑	5.3%↑	2.6%↑	6.1%↑	5.3%↑	-6.1%↓	-0.9%↓	-1.8%↓	-6.1%↓	6.1%↑	4.4%↑
Capital One	0.9%↑	-6.1%↓	-6.1%↓	-1.8%↓	2.6%↑	-0.9%↓	6.1%↑	5.3%↑	2.6%↑	6.1%↑	5.3%↑	-6.1%↓	-0.9%↓	-1.8%↓	-6.1%↓	6.1%↑	4.4%↑
Wells Fargo	0.9%↑	-6.1%↓	-6.1%↓	-1.8%↓	2.6%↑	-0.9%↓	6.1%↑	5.3%↑	2.6%↑	6.1%↑	5.3%↑	-6.1%↓	-0.9%↓	-1.8%↓	-6.1%↓	6.1%↑	4.4%↑
Navient	7.4%↑	0.8%↑	0.8%↑	4.9%↑	9%↑	5.7%↑	12.3%↑	11.5%↑	9%↑	12.3%↑	11.5%↑	0.8%↑	5.7%↑	4.9%↑	0.8%↑	12.3%↑	10.7%↑
Synchrony	5%↑	-1.7%↓	-1.7%↓	2.5%↑	6.7%↑	3.4%↑	10.1%↑	9.2%↑	6.7%↑	10.1%↑	9.2%↑	-1.7%↓	3.4%↑	2.5%↑	-1.7%↓	10.1%↑	8.4%↑
Amex	8.1%↑	1.6%↑	1.6%↑	5.7%↑	9.8%↑	6.5%↑	13%↑	12.2%↑	9.8%↑	13%↑	12.2%↑	1.6%↑	6.5%↑	5.7%↑	1.6%↑	13%↑	11.4%↑
U.S. Bank	7.4%↑	0.8%↑	0.8%↑	4.9%↑	9%↑	5.7%↑	12.3%↑	11.5%↑	9%↑	12.3%↑	11.5%↑	0.8%↑	5.7%↑	4.9%↑	0.8%↑	12.3%↑	10.7%↑
Portfolio Recovery	13.1%↑	6.9%↑	6.9%↑	10.8%↑	14.6%↑	11.5%↑	17.7%↑	16.9%↑	14.6%↑	17.7%↑	16.9%↑	6.9%↑	11.5%↑	10.8%↑	6.9%↑	17.7%↑	16.2%↑
PayPal	11%↑	4.7%↑	4.7%↑	8.7%↑	12.6%↑	9.4%↑	15.7%↑	15%↑	12.6%↑	15.7%↑	15%↑	4.7%↑	9.4%↑	8.7%↑	4.7%↑	15.7%↑	14.2%↑
Bread Financial	11%↑	4.7%↑	4.7%↑	8.7%↑	12.6%↑	9.4%↑	15.7%↑	15%↑	12.6%↑	15.7%↑	15%↑	4.7%↑	9.4%↑	8.7%↑	4.7%↑	15.7%↑	14.2%↑
Discover	9.6%↑	3.2%↑	3.2%↑	7.2%↑	11.2%↑	8%↑	14.4%↑	13.6%↑	11.2%↑	14.4%↑	13.6%↑	3.2%↑	8%↑	7.2%↑	3.2%↑	14.4%↑	12.8%↑
Nationstar	8.1%↑	1.6%↑	1.6%↑	5.7%↑	9.8%↑	6.5%↑	13%↑	12.2%↑	9.8%↑	13%↑	12.2%↑	1.6%↑	6.5%↑	5.7%↑	1.6%↑	13%↑	11.4%↑
AES	11%↑	4.7%↑	4.7%↑	8.7%↑	12.6%↑	9.4%↑	15.7%↑	15%↑	12.6%↑	15.7%↑	15%↑	4.7%↑	9.4%↑	8.7%↑	4.7%↑	15.7%↑	14.2%↑
Ocwen	8.9%↑	2.4%↑	2.4%↑	6.5%↑	10.5%↑	7.3%↑	13.7%↑	12.9%↑	10.5%↑	13.7%↑	12.9%↑	2.4%↑	7.3%↑	6.5%↑	2.4%↑	13.7%↑	12.1%↑
Encore Capital	13.7%↑	7.6%↑	7.6%↑	11.5%↑	15.3%↑	12.2%↑	18.3%↑	17.6%↑	15.3%↑	18.3%↑	17.6%↑	7.6%↑	12.2%↑	11.5%↑	7.6%↑	18.3%↑	16.8%↑
TD Bank	11%↑	4.7%↑	4.7%↑	8.7%↑	12.6%↑	9.4%↑	15.7%↑	15%↑	12.6%↑	15.7%↑	15%↑	4.7%↑	9.4%↑	8.7%↑	4.7%↑	15.7%↑	14.2%↑
PNC	11.7%↑	5.5%↑	5.5%↑	9.4%↑	13.3%↑	10.2%↑	16.4%↑	15.6%↑	13.3%↑	16.4%↑	15.6%↑	5.5%↑	10.2%↑	9.4%↑	5.5%↑	16.4%↑	14.8%↑
Barclays	12.4%↑	6.2%↑	6.2%↑	10.1%↑	14%↑	10.9%↑	17.1%↑	16.3%↑	14%↑	17.1%↑	16.3%↑	6.2%↑	10.9%↑	10.1%↑	6.2%↑	17.1%↑	15.5%↑
Santander	13.7%↑	7.6%↑	7.6%↑	11.5%↑	15.3%↑	12.2%↑	18.3%↑	17.6%↑	15.3%↑	18.3%↑	17.6%↑	7.6%↑	12.2%↑	11.5%↑	7.6%↑	18.3%↑	16.8%↑
Ally	15%↑	9%↑	9%↑	12.8%↑	16.5%↑	13.5%↑	19.5%↑	18.8%↑	16.5%↑	19.5%↑	18.8%↑	9%↑	13.5%↑	12.8%↑	9%↑	19.5%↑	18%↑
Resurgent Capital	16.9%↑	11%↑	11%↑	14.7%↑	18.4%↑	15.4%↑	21.3%↑	20.6%↑	18.4%↑	21.3%↑	20.6%↑	11%↑	15.4%↑	14.7%↑	11%↑	21.3%↑	19.9%↑
USAA	13.1%↑	6.9%↑	6.9%↑	10.8%↑	14.6%↑	11.5%↑	17.7%↑	16.9%↑	14.6%↑	17.7%↑	16.9%↑	6.9%↑	11.5%↑	10.8%↑	6.9%↑	17.7%↑	16.2%↑
ERC	18.1%↑	12.3%↑	12.3%↑	15.9%↑	19.6%↑	16.7%↑	22.5%↑	21.7%↑	19.6%↑	22.5%↑	21.7%↑	12.3%↑	16.7%↑	15.9%↑	12.3%↑	22.5%↑	21%↑
Navy Federal	13.7%↑	7.6%↑	7.6%↑	11.5%↑	15.3%↑	12.2%↑	18.3%↑	17.6%↑	15.3%↑	18.3%↑	17.6%↑	7.6%↑	12.2%↑	11.5%↑	7.6%↑	18.3%↑	16.8%↑
Coinbase	20.4%↑	14.8%↑	14.8%↑	18.3%↑	21.8%↑	19%↑	24.6%↑	23.9%↑	21.8%↑	24.6%↑	23.9%↑	14.8%↑	19%↑	18.3%↑	14.8%↑	24.6%↑	23.2%↑

Table A.2: Comparative Analysis of Tokenizer Fertility Improvement (φ) at the Vocabulary Level Across Local, Pre-trained, and Federated Approaches

	BERT	BART	GPT ₂	GPT _{3.5} -Turbo	T ₅	Llama2	SecBERT	FinBERT	FinancialBERT	FLANG-BERT	FinanceDeBERTa	FinanceDistilGPT ₂	FinGPT	FinMegatronBERT	FinMegatronGPT	Centralized	Federated
Equifax	-4.8%↓	-2.9%↓	-2.9%↓	5.3%↑	3.4%↑	-2.4%↓	6.3%↑	8.2%↑	9.6%↑	8.7%↑	8.2%↑	-2.9%↓	-2.4%↓	-1.4%↓	-2.9%↓	2.4%↑	1.9%↑
Experian	-4.8%↓	-2.9%↓	-2.9%↓	5.3%↑	3.4%↑	-2.4%↓	6.3%↑	8.2%↑	9.6%↑	8.7%↑	8.2%↑	-2.9%↓	-2.4%↓	-1.4%↓	-2.9%↓	2.4%↑	1.9%↑
TransUnion	-3.8%↓	-1.9%↓	-1.9%↓	6.2%↑	4.3%↑	-1.4%↓	7.1%↑	9.0%↑	10.5%↑	9.5%↑	9.0%↑	-1.9%↓	-1.4%↓	0%	-1.9%↓	3.3%↑	2.9%↑
Bank of America	-2.3%↓	-0.5%↓	-0.5%↓	7.5%↑	5.6%↑	0%	8.5%↑	10.3%↑	11.7%↑	10.8%↑	10.3%↑	-0.5%↓	0%	0.9%↑	-0.5%↓	4.7%↑	4.2%↑
JPMorgan Chase	-2.3%↓	-0.5%↓	-0.5%↓	7.5%↑	5.6%↑	0%	8.5%↑	10.3%↑	11.7%↑	10.8%↑	10.3%↑	-0.5%↓	0%	0.9%↑	-0.5%↓	4.7%↑	4.2%↑
Citibank	-1.4%↓	0.5%↑	0.5%↑	8.4%↑	6.5%↑	0.9%↑	9.3%↑	11.2%↑	12.6%↑	11.6%↑	11.2%↑	0.5%↑	0.9%↑	1.9%↑	0.5%↑	5.6%↑	5.1%↑
Capital One	0%	1.8%↑	1.8%↑	9.6%↑	7.8%↑	2.3%↑	10.6%↑	12.4%↑	13.8%↑	12.8%↑	12.4%↑	1.8%↑	2.3%↑	3.2%↑	1.8%↑	6.9%↑	6.4%↑
Wells Fargo	-2.3%↓	-0.5%↓	-0.5%↓	7.5%↑	5.6%↑	0%	8.5%↑	10.3%↑	11.7%↑	10.8%↑	10.3%↑	-0.5%↓	0%	0.9%↑	-0.5%↓	4.7%↑	4.2%↑
Navient	6.0%↑	7.8%↑	7.8%↑	15.1%↑	13.4%↑	8.2%↑	15.9%↑	17.7%↑	19.0%↑	18.1%↑	17.7%↑	7.8%↑	8.2%↑	9.1%↑	7.8%↑	12.5%↑	12.1%↑
Synchrony	4.8%↑	6.6%↑	6.6%↑	14.0%↑	12.2%↑	7.0%↑	14.8%↑	16.6%↑	17.9%↑	17.0%↑	16.6%↑	6.6%↑	7.0%↑	7.9%↑	6.6%↑	11.4%↑	10.9%↑
Amex	7.6%↑	9.3%↑	9.3%↑	16.5%↑	14.8%↑	9.7%↑	17.4%↑	19.1%↑	20.3%↑	19.5%↑	19.1%↑	9.3%↑	9.7%↑	10.6%↑	9.3%↑	14.0%↑	13.6%↑
U.S. Bank	6.8%↑	8.5%↑	8.5%↑	15.8%↑	14.1%↑	9.0%↑	16.7%↑	18.4%↑	19.7%↑	18.8%↑	18.4%↑	8.5%↑	9.0%↑	9.8%↑	8.5%↑	13.2%↑	12.8%↑
Portfolio Recovery	14.2%↑	15.7%↑	15.7%↑	22.4%↑	20.9%↑	16.1%↑	23.2%↑	24.8%↑	26.0%↑	25.2%↑	24.8%↑	15.7%↑	16.1%↑	16.9%↑	15.7%↑	20.1%↑	19.7%↑
PayPal	11.0%↑	12.7%↑	12.7%↑	19.6%↑	18.0%↑	13.1%↑	20.4%↑	22.0%↑	23.3%↑	22.4%↑	22.0%↑	12.7%↑	13.1%↑	13.9%↑	12.7%↑	17.1%↑	16.7%↑
Bread Financial	12.1%↑	13.7%↑	13.7%↑	20.6%↑	19.0%↑	14.1%↑	21.4%↑	23.0%↑	24.2%↑	23.4%↑	23.0%↑	13.7%↑	14.1%↑	14.9%↑	13.7%↑	18.1%↑	17.7%↑
Discover	8.8%↑	10.5%↑	10.5%↑	17.6%↑	15.9%↑	10.9%↑	18.4%↑	20.1%↑	21.3%↑	20.5%↑	20.1%↑	10.5%↑	10.9%↑	11.7%↑	10.5%↑	15.1%↑	14.6%↑
Nationstar	7.2%↑	8.9%↑	8.9%↑	16.2%↑	14.5%↑	9.4%↑	17.0%↑	18.7%↑	20.0%↑	19.1%↑	18.7%↑	8.9%↑	9.4%↑	10.2%↑	8.9%↑	13.6%↑	13.2%↑
AES	11.7%↑	13.4%↑	13.4%↑	20.2%↑	18.6%↑	13.8%↑	21.1%↑	22.7%↑	23.9%↑	23.1%↑	22.7%↑	13.4%↑	13.8%↑	14.6%↑	13.4%↑	17.8%↑	17.4%↑
Ocwen	7.6%↑	9.3%↑	9.3%↑	16.5%↑	14.8%↑	9.7%↑	17.4%↑	19.1%↑	20.3%↑	19.5%↑	19.1%↑	9.3%↑	9.7%↑	10.6%↑	9.3%↑	14.0%↑	13.6%↑
Encore Capital	14.5%↑	16.1%↑	16.1%↑	22.7%↑	21.2%↑	16.5%↑	23.5%↑	25.1%↑	26.3%↑	25.5%↑	25.1%↑	16.1%↑	16.5%↑	17.3%↑	16.1%↑	20.4%↑	20.0%↑
TD Bank	11.0%↑	12.7%↑	12.7%↑	19.6%↑	18.0%↑	13.1%↑	20.4%↑	22.0%↑	23.3%↑	22.4%↑	22.0%↑	12.7%↑	13.1%↑	13.9%↑	12.7%↑	17.1%↑	16.7%↑
PNC	11.7%↑	13.4%↑	13.4%↑	20.2%↑	18.6%↑	13.8%↑	21.1%↑	22.7%↑	23.9%↑	23.1%↑	22.7%↑	13.4%↑	13.8%↑	14.6%↑	13.4%↑	17.8%↑	17.4%↑
Barclays	13.1%↑	14.7%↑	14.7%↑	21.5%↑	19.9%↑	15.1%↑	22.3%↑	23.9%↑	25.1%↑	24.3%↑	23.9%↑	14.7%↑	15.1%↑	15.9%↑	14.7%↑	19.1%↑	18.7%↑
Santander	14.2%↑	15.7%↑	15.7%↑	22.4%↑	20.9%↑	16.1%↑	23.2%↑	24.8%↑	26.0%↑	25.2%↑	24.8%↑	15.7%↑	16.1%↑	16.9%↑	15.7%↑	20.1%↑	19.7%↑
Ally	14.5%↑	16.1%↑	16.1%↑	22.7%↑	21.2%↑	16.5%↑	23.5%↑	25.1%↑	26.3%↑	25.5%↑	25.1%↑	16.1%↑	16.5%↑	17.3%↑	16.1%↑	20.4%↑	20.0%↑
Resurgent Capital	18.0%↑	19.5%↑	19.5%↑	25.9%↑	24.4%↑	19.9%↑	26.7%↑	28.2%↑	29.3%↑	28.6%↑	28.2%↑	19.5%↑	19.9%↑	20.7%↑	19.5%↑	23.7%↑	23.3%↑
USAA	12.4%↑	14.1%↑	14.1%↑	20.9%↑	19.3%↑	14.5%↑	21.7%↑	23.3%↑	24.5%↑	23.7%↑	23.3%↑	14.1%↑	14.5%↑	15.3%↑	14.1%↑	18.5%↑	18.1%↑
ERC	19.3%↑	20.7%↑	20.7%↑	27.0%↑	25.6%↑	21.1%↑	27.8%↑	29.3%↑	30.4%↑	29.6%↑	29.3%↑	20.7%↑	21.1%↑	21.9%↑	20.7%↑	24.8%↑	24.4%↑
Navy Federal	13.1%↑	14.7%↑	14.7%↑	21.5%↑	19.9%↑	15.1%↑	22.3%↑	23.9%↑	25.1%↑	24.3%↑	23.9%↑	14.7%↑	15.1%↑	15.9%↑	14.7%↑	19.1%↑	18.7%↑
Coinbase	19.0%↑	20.4%↑	20.4%↑	26.8%↑	25.3%↑	20.8%↑	27.5%↑	29.0%↑	30.1%↑	29.4%↑	29.0%↑	20.4%↑	20.8%↑	21.6%↑	20.4%↑	24.5%↑	24.2%↑

Table A.3: Comparative Analysis of Tokenizer Proportion of Continued Words Improvement (ψ) at the Document Level Across Local, Pre-trained, and Federated Approaches

	BERT	BART	GPT ₂	GPT _{3.5} -Turbo	T ₅	Llama2	SecBERT	FinBERT	FinancialBERT	FLANG-BERT	FinanceDeBERTa	FinanceDistilGPT ₂	FinGPT	FinMegatronBERT	FinMegatronGPT	Centralized	Federated
Equifax	0%	-70%↓	-70%↓	-30%↓	0%	-20%↓	20%↑	30%↑	10%↑	30%↑	40%↑	-70%↓	-20%↓	-50%↓	-70%↓	40%↑	20%↑
Experian	0%	-70%↓	-70%↓	-30%↓	0%	-20%↓	20%↑	30%↑	10%↑	30%↑	40%↑	-70%↓	-20%↓	-50%↓	-70%↓	40%↑	20%↑
TransUnion	9.1%↑	-54.5%↓	-54.5%↓	-18.2%↓	9.1%↑	-9.1%↓	27.3%↑	36.4%↑	18.2%↑	36.4%↑	45.5%↑	-54.5%↓	-9.1%↓	-36.4%↓	-54.5%↓	45.5%↑	27.3%↑
Bank of America	16.7%↑	-41.7%↓	-41.7%↓	-8.3%↓	16.7%↑	0%	33.3%↑	41.7%↑	25%↑	41.7%↑	50%↑	-41.7%↓	0%	-25%↓	-41.7%↓	50%↑	33.3%↑
JPMorgan Chase	16.7%↑	-41.7%↓	-41.7%↓	-8.3%↓	16.7%↑	0%	33.3%↑	41.7%↑	25%↑	41.7%↑	50%↑	-41.7%↓	0%	-25%↓	-41.7%↓	50%↑	33.3%↑
Citibank	16.7%↑	-41.7%↓	-41.7%↓	-8.3%↓	16.7%↑	0%	33.3%↑	41.7%↑	25%↑	41.7%↑	50%↑	-41.7%↓	0%	-25%↓	-41.7%↓	50%↑	33.3%↑
Capital One	16.7%↑	-41.7%↓	-41.7%↓	-8.3%↓	16.7%↑	0%	33.3%↑	41.7%↑	25%↑	41.7%↑	50%↑	-41.7%↓	0%	-25%↓	-41.7%↓	50%↑	33.3%↑
Wells Fargo	16.7%↑	-41.7%↓	-41.7%↓	-8.3%↓	16.7%↑	0%	33.3%↑	41.7%↑	25%↑	41.7%↑	50%↑	-41.7%↓	0%	-25%↓	-41.7%↓	50%↑	33.3%↑
Navient	41.2%↑	0%	0%	23.5%↑	41.2%↑	29.4%↑	52.9%↑	58.8%↑	47.1%↑	58.8%↑	64.7%↑	0%	29.4%↑	11.8%↑	0%	64.7%↑	52.9%↑
Synchrony	33.3%↑	-13.3%↓	-13.3%↓	13.3%↑	33.3%↑	20%↑	46.7%↑	53.3%↑	40%↑	53.3%↑	60%↑	-13.3%↓	20%↑	0%	-13.3%↓	60%↑	46.7%↑
Amex	44.4%↑	5.6%↑	5.6%↑	27.8%↑	44.4%↑	33.3%↑	55.6%↑	61.1%↑	50%↑	61.1%↑	66.7%↑	5.6%↑	33.3%↑	16.7%↑	5.6%↑	66.7%↑	55.6%↑
U.S. Bank	41.2%↑	0%	0%	23.5%↑	41.2%↑	29.4%↑	52.9%↑	58.8%↑	47.1%↑	58.8%↑	64.7%↑	0%	29.4%↑	11.8%↑	0%	64.7%↑	52.9%↑
Portfolio Recovery	56.5%↑	26.1%↑	26.1%↑	43.5%↑	56.5%↑	47.8%↑	65.2%↑	69.6%↑	60.9%↑	69.6%↑	73.9%↑	26.1%↑	47.8%↑	34.8%↑	26.1%↑	73.9%↑	65.2%↑
PayPal	52.4%↑	19%↑	19%↑	38.1%↑	52.4%↑	42.9%↑	61.9%↑	66.7%↑	57.1%↑	66.7%↑	71.4%↑	19%↑	42.9%↑	28.6%↑	19%↑	71.4%↑	61.9%↑
Bread Financial	52.4%↑	19%↑	19%↑	38.1%↑	52.4%↑	42.9%↑	61.9%↑	66.7%↑	57.1%↑	66.7%↑	71.4%↑	19%↑	42.9%↑	28.6%↑	19%↑	71.4%↑	61.9%↑
Discover	47.4%↑	10.5%↑	10.5%↑	31.6%↑	47.4%↑	36.8%↑	57.9%↑	63.2%↑	52.6%↑	63.2%↑	68.4%↑	10.5%↑	36.8%↑	21.1%↑	10.5%↑	68.4%↑	57.9%↑
Nationstar	47.4%↑	10.5%↑	10.5%↑	31.6%↑	47.4%↑	36.8%↑	57.9%↑	63.2%↑	52.6%↑	63.2%↑	68.4%↑	10.5%↑	36.8%↑	21.1%↑	10.5%↑	68.4%↑	57.9%↑
AES	52.4%↑	19%↑	19%↑	38.1%↑	52.4%↑	42.9%↑	61.9%↑	66.7%↑	57.1%↑	66.7%↑	71.4%↑	19%↑	42.9%↑	28.6%↑	19%↑	71.4%↑	61.9%↑
Ocwen	47.4%↑	10.5%↑	10.5%↑	31.6%↑	47.4%↑	36.8%↑	57.9%↑	63.2%↑	52.6%↑	63.2%↑	68.4%↑	10.5%↑	36.8%↑	21.1%↑	10.5%↑	68.4%↑	57.9%↑
Encore Capital	56.5%↑	26.1%↑	26.1%↑	43.5%↑	56.5%↑	47.8%↑	65.2%↑	69.6%↑	60.9%↑	69.6%↑	73.9%↑	26.1%↑	47.8%↑	34.8%↑	26.1%↑	73.9%↑	65.2%↑
TD Bank	52.4%↑	19%↑	19%↑	38.1%↑	52.4%↑	42.9%↑	61.9%↑	66.7%↑	57.1%↑	66.7%↑	71.4%↑	19%↑	42.9%↑	28.6%↑	19%↑	71.4%↑	61.9%↑
PNC	52.4%↑	19%↑	19%↑	38.1%↑	52.4%↑	42.9%↑	61.9%↑	66.7%↑	57.1%↑	66.7%↑	71.4%↑	19%↑	42.9%↑	28.6%↑	19%↑	71.4%↑	61.9%↑
Barclays	54.5%↑	22.7%↑	22.7%↑	40.9%↑	54.5%↑	45.5%↑	63.6%↑	68.2%↑	59.1%↑	68.2%↑	72.7%↑	22.7%↑	45.5%↑	31.8%↑	22.7%↑	72.7%↑	63.6%↑
Santander	56.5%↑	26.1%↑	26.1%↑	43.5%↑	56.5%↑	47.8%↑	65.2%↑	69.6%↑	60.9%↑	69.6%↑	73.9%↑	26.1%↑	47.8%↑	34.8%↑	26.1%↑	73.9%↑	65.2%↑
Ally	58.3%↑	29.2%↑	29.2%↑	45.8%↑	58.3%↑	50%↑	66.7%↑	70.8%↑	62.5%↑	70.8%↑	75%↑	29.2%↑	50%↑	37.5%↑	29.2%↑	75%↑	66.7%↑
Resurgent Capital	61.5%↑	34.6%↑	34.6%↑	50%↑	61.5%↑	53.8%↑	69.2%↑	73.1%↑	65.4%↑	73.1%↑	76.9%↑	34.6%↑	53.8%↑	42.3%↑	34.6%↑	76.9%↑	69.2%↑
USAA	56.5%↑	26.1%↑	26.1%↑	43.5%↑	56.5%↑	47.8%↑	65.2%↑	69.6%↑	60.9%↑	69.6%↑	73.9%↑	26.1%↑	47.8%↑	34.8%↑	26.1%↑	73.9%↑	65.2%↑
ERC	63%↑	37%↑	37%↑	51.9%↑	63%↑	55.6%↑	70.4%↑	74.1%↑	66.7%↑	74.1%↑	77.8%↑	37%↑	55.6%↑	44.4%↑	37%↑	77.8%↑	70.4%↑
Navy Federal	56.5%↑	26.1%↑	26.1%↑	43.5%↑	56.5%↑	47.8%↑	65.2%↑	69.6%↑	60.9%↑	69.6%↑	73.9%↑	26.1%↑	47.8%↑	34.8%↑	26.1%↑	73.9%↑	65.2%↑
Coinbase	65.5%↑	41.4%↑	41.4%↑	55.2%↑	65.5%↑	58.6%↑	72.4%↑	75.9%↑	69%↑	75.9%↑	79.3%↑	41.4%↑	58.6%↑	48.3%↑	41.4%↑	79.3%↑	72.4%↑

Table A.4: Comparative Analysis of Tokenizer Proportion of Continued Words Improvement (ψ) at the Vocabulary Level Across Local, Pre-trained, and Federated Approaches

	BERT	BART	GPT ₂	GPT _{3.5} -Turbo	T ₅	Llama2	SecBERT	FinBERT	FinancialBERT	FLANG-BERT	FinanceDeBERTa	FinanceDistilGPT ₂	FinGPT	FinMegatronBERT	FinMegatronGPT	Centralized	Federated
Equifax	-3.8%↓	-1.9%↓	-1.9%↓	5.8%↑	3.8%↑	-1.9%↓	30.8%↑	42.3%↑	15.4%↑	28.8%↑	38.5%↑	-1.9%↓	-1.9%↓	9.6%↑	-1.9%↓	1.9%↑	1.9%↑
Experian	-3.8%↓	-1.9%↓	-1.9%↓	5.8%↑	3.8%↑	-1.9%↓	30.8%↑	42.3%↑	15.4%↑	28.8%↑	38.5%↑	-1.9%↓	-1.9%↓	9.6%↑	-1.9%↓	1.9%↑	1.9%↑
TransUnion	-3.8%↓	-1.9%↓	-1.9%↓	5.8%↑	3.8%↑	-1.9%↓	30.8%↑	42.3%↑	15.4%↑	28.8%↑	38.5%↑	-1.9%↓	-1.9%↓	9.6%↑	-1.9%↓	1.9%↑	1.9%↑
Bank of America	-1.9%↓	0%	0%	7.5%↑	5.7%↑	0%	32.1%↑	43.4%↑	17%↑	30.2%↑	39.6%↑	0%	0%	11.3%↑	0%	3.8%↑	3.8%↑
JPMorgan Chase	-1.9%↓	0%	0%	7.5%↑	5.7%↑	0%	32.1%↑	43.4%↑	17%↑	30.2%↑	39.6%↑	0%	0%	11.3%↑	0%	3.8%↑	3.8%↑
Citibank	-1.9%↓	0%	0%	7.5%↑	5.7%↑	0%	32.1%↑	43.4%↑	17%↑	30.2%↑	39.6%↑	0%	0%	11.3%↑	0%	3.8%↑	3.8%↑
Capital One	0%	1.9%↑	1.9%↑	9.3%↑	7.4%↑	1.9%↑	33.3%↑	44.4%↑	18.5%↑	31.1%↑	40.7%↑	1.9%↑	1.9%↑	13%↑	1.9%↑	5.6%↑	5.6%↑
Wells Fargo	-1.9%↓	0%	0%	7.5%↑	5.7%↑	0%	32.1%↑	43.4%↑	17%↑	30.2%↑	39.6%↑	0%	0%	11.3%↑	0%	3.8%↑	3.8%↑
Navient	5.3%↑	7%↑	7%↑	14%↑	12.3%↑	7%↑	36.8%↑	47.4%↑	22.8%↑	35.1%↑	43.9%↑	7%↑	7%↑	17.5%↑	7%↑	10.5%↑	10.5%↑
Synchrony	3.6%↑	5.4%↑	5.4%↑	12.5%↑	10.7%↑	5.4%↑	35.7%↑	46.4%↑	21.4%↑	33.9%↑	42.9%↑	5.4%↑	5.4%↑	16.1%↑	5.4%↑	8.9%↑	8.9%↑
Amex	5.3%↑	7%↑	7%↑	14%↑	12.3%↑	7%↑	36.8%↑	47.4%↑	22.8%↑	35.1%↑	43.9%↑	7%↑	7%↑	17.5%↑	7%↑	10.5%↑	10.5%↑
U.S. Bank	5.3%↑	7%↑	7%↑	14%↑	12.3%↑	7%↑	36.8%↑	47.4%↑	22.8%↑	35.1%↑	43.9%↑	7%↑	7%↑	17.5%↑	7%↑	10.5%↑	10.5%↑
Portfolio Recovery	10%↑	11.7%↑	11.7%↑	18.3%↑	16.7%↑	11.7%↑	40%↑	50%↑	26.7%↑	38.3%↑	46.7%↑	11.7%↑	11.7%↑	21.7%↑	11.7%↑	15%↑	15%↑
PayPal	8.5%↑	10.2%↑	10.2%↑	16.9%↑	15.3%↑	10.2%↑	39%↑	49.2%↑	25.4%↑	37.3%↑	45.8%↑	10.2%↑	10.2%↑	20.3%↑	10.2%↑	13.6%↑	13.6%↑
Bread Financial	10%↑	11.7%↑	11.7%↑	18.3%↑	16.7%↑	11.7%↑	40%↑	50%↑	26.7%↑	38.3%↑	46.7%↑	11.7%↑	11.7%↑	21.7%↑	11.7%↑	15%↑	15%↑
Discover	6.9%↑	8.6%↑	8.6%↑	15.5%↑	13.8%↑	8.6%↑	37.9%↑	48.3%↑	24.1%↑	36.2%↑	44.8%↑	8.6%↑	8.6%↑	19%↑	8.6%↑	12.1%↑	12.1%↑
Nationstar	5.3%↑	7%↑	7%↑	14%↑	12.3%↑	7%↑	36.8%↑	47.4%↑	22.8%↑	35.1%↑	43.9%↑	7%↑	7%↑	17.5%↑	7%↑	10.5%↑	10.5%↑
AES	8.5%↑	10.2%↑	10.2%↑	16.9%↑	15.3%↑	10.2%↑	39%↑	49.2%↑	25.4%↑	37.3%↑	45.8%↑	10.2%↑	10.2%↑	20.3%↑	10.2%↑	13.6%↑	13.6%↑
Ocwen	6.9%↑	8.6%↑	8.6%↑	15.5%↑	13.8%↑	8.6%↑	37.9%↑	48.3%↑	24.1%↑	36.2%↑	44.8%↑	8.6%↑	8.6%↑	19%↑	8.6%↑	12.1%↑	12.1%↑
Encore Capital	11.5%↑	13.1%↑	13.1%↑	19.7%↑	18%↑	13.1%↑	41%↑	50.8%↑	27.9%↑	39.3%↑	47.5%↑	13.1%↑	13.1%↑	23%↑	13.1%↑	16.4%↑	16.4%↑
TD Bank	8.5%↑	10.2%↑	10.2%↑	16.9%↑	15.3%↑	10.2%↑	39%↑	49.2%↑	25.4%↑	37.3%↑	45.8%↑	10.2%↑	10.2%↑	20.3%↑	10.2%↑	13.6%↑	13.6%↑
PNC	8.5%↑	10.2%↑	10.2%↑	16.9%↑	15.3%↑	10.2%↑	39%↑	49.2%↑	25.4%↑	37.3%↑	45.8%↑	10.2%↑	10.2%↑	20.3%↑	10.2%↑	13.6%↑	13.6%↑
Barclays	10%↑	11.7%↑	11.7%↑	18.3%↑	16.7%↑	11.7%↑	40%↑	50%↑	26.7%↑	38.3%↑	46.7%↑	11.7%↑	11.7%↑	21.7%↑	11.7%↑	15%↑	15%↑
Santander	11.5%↑	13.1%↑	13.1%↑	19.7%↑	18%↑	13.1%↑	41%↑	50.8%↑	27.9%↑	39.3%↑	47.5%↑	13.1%↑	13.1%↑	23%↑	13.1%↑	16.4%↑	16.4%↑
Ally	11.5%↑	13.1%↑	13.1%↑	19.7%↑	18%↑	13.1%↑	41%↑	50.8%↑	27.9%↑	39.3%↑	47.5%↑	13.1%↑	13.1%↑	23%↑	13.1%↑	16.4%↑	16.4%↑
Resurgent Capital	12.9%↑	14.5%↑	14.5%↑	21%↑	19.4%↑	14.5%↑	41.9%↑	51.6%↑	29%↑	40.3%↑	48.4%↑	14.5%↑	14.5%↑	24.2%↑	14.5%↑	17.7%↑	17.7%↑
USAA	10%↑	11.7%↑	11.7%↑	18.3%↑	16.7%↑	11.7%↑	40%↑	50%↑	26.7%↑	38.3%↑	46.7%↑	11.7%↑	11.7%↑	21.7%↑	11.7%↑	15%↑	15%↑
ERC	14.3%↑	15.9%↑	15.9%↑	22.2%↑	20.6%↑	15.9%↑	42.9%↑	52.4%↑	30.2%↑	41.3%↑	49.2%↑	15.9%↑	15.9%↑	25.4%↑	15.9%↑	19%↑	19%↑
Navy Federal	10%↑	11.7%↑	11.7%↑	18.3%↑	16.7%↑	11.7%↑	40%↑	50%↑	26.7%↑	38.3%↑	46.7%↑	11.7%↑	11.7%↑	21.7%↑	11.7%↑	15%↑	15%↑
Coinbase	14.3%↑	15.9%↑	15.9%↑	22.2%↑	20.6%↑	15.9%↑	42.9%↑	52.4%↑	30.2%↑	41.3%↑	49.2%↑	15.9%↑	15.9%↑	25.4%↑	15.9%↑	19%↑	19%↑

Bibliography

- [AFG16] Myrto Arapinis, Diego Figueira, and Marco Gaboardi. Sensitivity of counting queries. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2016.
- [AGM⁺22] Ehsan Amid, Arun Ganesh, Rajiv Mathews, Swaroop Ramaswamy, Shuang Song, Thomas Steinke, Vinith M Suriyakumar, Om Thakkar, and Abhradeep Thakurta. Public data-assisted mirror descent for private model training. In *International Conference on Machine Learning*, pages 517–535. PMLR, 2022.
- [AKA17] Kumar Anand, Jyoti Kumar, and Kumar Anand. Anomaly detection in online social network: a survey. In *ICICCT 2017*, pages 456–459. IEEE, 2017.
- [AMF10] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*, pages 410–421, 2010.
- [AMZ16] Abdalhamid Abdallah, Mohd Afizi Maarof, and Azween Zainal. Fraud detection system: a survey. *Journal of Network and Computer Applications*, 68:90–113, 2016.
- [Ana16] McKinsey Analytics. The age of analytics: competing in a data-driven world. *McKinsey Global Institute Research*, 2016.
- [Ano22a] Anonymous. Grouping-matrix based graph pooling with adaptive number of clusters. *arXiv preprint arXiv:2209.02939*, 2022. <https://arxiv.labs.arxiv.org/html/2209.02939>.
- [Ano22b] Anonymous. Higher-order clustering and pooling for graph neural networks. *arXiv preprint arXiv:2209.03473*, 2022. <https://arxiv.labs.arxiv.org/html/2209.03473>.
- [ANRD15] Nesreen K Ahmed, Jennifer Neville, Ryan A Rossi, and Nick Duffield. Efficient graphlet counting for large networks. In *2015 IEEE international conference on data mining*, pages 1–10. IEEE, 2015.

- [AS14] Charu C. Aggarwal and Karthik Subbian. Evolutionary network analysis: A survey. *ACM Computing Surveys*, 47(1):10:1–10:36, 2014.
- [ASFEE24] Mustafa Abdul Salam, Khaled M Fouad, Doaa L Elbably, and Salah M Elsayed. Federated learning model for credit card fraud detection with data balancing techniques. *Neural Computing and Applications*, 36(11):6231–6256, 2024.
- [ATK15] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2015.
- [Ban14] European Central Bank. Technical report. Technical report, European Central Bank, 2014.
- [BBG18] Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. *Advances in neural information processing systems*, 31, 2018.
- [BBG20] Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy profiles and amplification by subsampling. *Journal of Privacy and Confidentiality*, 10(1), 2020.
- [BDJV22] Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. Lamp: Extracting text from gradients with language model priors. *Advances in Neural Information Processing Systems*, 35:7641–7654, 2022.
- [BDQ⁺19] Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. Unsupervised inductive graph-level representation learning via graph-graph proximity. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 1988–1994, 2019.
- [BDV00] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.
- [BGBV22] Alexandra Benamar, Cyril Grouin, Meryl Bothua, and Anne Vilnat. Evaluating tokenizers impact on oovs representation with transformers models. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4193–4204, 2022.
- [BHo1] Richard J. Bolton and David J. Hand. Unsupervised profiling methods for fraud detection. In *Proceedings of Conference Credit Scoring and Credit Control VII*, pages 5–7, 2001.

- [BHB⁺18] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.
- [BIK⁺17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Van Overveldt, Adriana Chien, and Marco Corso. Practical secure aggregation for privacy-preserving machine learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.
- [BJTW11] Somnath Bhattacharyya, Sanjay Jha, K Tharakunnel, and JC Westland. Data mining for credit card fraud: a comparative study. *Decision Support Systems*, 50(3):602–613, 2011.
- [BK05] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pages 8–pp. IEEE, 2005.
- [BN03] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.
- [BN21] Laura Blattner and Scott Nelson. How costly is noise? data and disparities in consumer credit. *arXiv preprint arXiv:2105.07554*, 2021.
- [Bon87] Phillip Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.
- [BPD03] Tej Paul Bhatla, Vikram Prabhu, and Amit Dua. Understanding credit card frauds. *Cards business review*, 1(6):1–15, 2003.
- [BPL⁺16] Peter W Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4502–4510, 2016.
- [BRNM21] Priyam Basu, Tiasa Singha Roy, Rakshit Naidu, and Zumrut Muftuoglu. Privacy enabled financial text classification using differential privacy and federated learning. *arXiv preprint arXiv:2110.01643*, 2021.
- [BSvD⁺22] Eugene Bagdasaryan, Congzheng Song, Rogier van Dalen, Matt Seigel, and Áine Cahill. Training a tokenizer for free with private federated learning. In *ACL*, 2022.

- [BvdM23] Martin Berglund and Brink van der Merwe. Formalizing bpe tokenization. *arXiv preprint arXiv:2309.08715*, 2023.
- [CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: a survey. *ACM Computing Surveys*, 41(3):1–58, 2009.
- [CCG24] E Chen, Yang Cao, and Yifei Ge. A generalized shuffle framework for privacy amplification: Strengthening privacy guarantees and enhancing utility. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11267–11275, 2024.
- [CD16] Victor Costan and Srinivas Devadas. Intel sgx explained. In *IACR Cryptology ePrint Archive*, volume 2016, page 086, 2016.
- [CGCB14] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [Chu97] Fan R. K. Chung. *Spectral Graph Theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1997.
- [CLQ⁺16] Rui Chen, Haoran Li, A Kai Qin, Shiva Prasad Kasiviswanathan, and Hongxia Jin. Private spatial data aggregation in the local setting. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 289–300. IEEE, 2016.
- [CND⁺23] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Sebastian Chung, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.
- [Com23] Commission de Surveillance du Secteur Financier (CSSF). Thematic review on the use of artificial intelligence in the luxembourg financial sector. Technical report, Commission de Surveillance du Secteur Financier, May 2023. Available online: <https://www.cssf.lu/en/document/thematic-review-on-the-use-of-artificial-intelligence-in-the-luxembourg>
- [CWH18] Yingmei Chen, Zhongyu Wei, and Xuanjing Huang. Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 1655–1658, 2018.
- [CWW⁺23] Dongqi Cai, Yaozong Wu, Shangguang Wang, Felix Xiaozhu Lin, and Mengwei Xu. Efficient federated learning for modern nlp. In *Proceedings*

- of the 29th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '23, New York, NY, USA, 2023. Association for Computing Machinery.
- [CZC₁₈] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE transactions on knowledge and data engineering*, 30(9):1616–1637, 2018.
- [DBV₁₆] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.
- [DCLOT⁺₁₈] Riccardo Di Clemente, Miguel Luengo-Oroz, Matias Travizano, Sharon Xu, Bapu Vaitla, and Marta C González. Sequences of purchases in credit card data reveal lifestyles in urban populations. *Nature communications*, 9(1):3330, 2018.
- [DCLT₁₉] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.
- [DCS₁₇] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144, 2017.
- [DELS₀₀] Curtis E. Dyreson, William S. Evans, Hong Lin, and Richard T. Snodgrass. Efficiently supporting temporal granularities. *IEEE Transactions on Knowledge and Data Engineering*, 12(4):568–587, 2000.
- [Den₈₀] Dorothy E Denning. Secure statistical databases with random sample queries. *ACM Transactions on Database Systems (TODS)*, 5(3):291–315, 1980.
- [DJW₁₃] John C Duchi, Michael I Jordan, and Martin J Wainwright. Local privacy and statistical minimax rates. In *54th Annual IEEE Symposium on Foundations of Computer Science*, pages 429–438. IEEE, 2013.
- [DMNS₀₆] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer, 2006.

- [DR⁺₁₄] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.
- [DRD₁₉] Shuoyang Ding, Adithya Renduchintala, and Kevin Duh. A call for prudent choice of subword merge operations in neural machine translation. *arXiv preprint arXiv:1905.10453*, 2019.
- [DRV₁₀] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st annual symposium on foundations of computer science*, pages 51–60. IEEE, 2010.
- [DSHS_{23a}] Farouk Damoun, Hamida Seba, Jean Hilger, and Radu State. G-hin2vec: Distributed heterogeneous graph representations for cardholder transactions. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, pages 528–535, 2023.
- [DSHS_{23b}] Farouk Damoun, Hamida Seba, Jean Hilger, and Radu State. Graph-level heterogeneous information network embeddings for cardholder transaction analysis. *Neural Computing and Applications*, pages 12–35, 2023.
- [DSR₂₄] Gautier Dagan, Gabriele Synnaeve, and Baptiste Rozière. Getting the most out of your tokenizer for pre-training and domain adaptation. *arXiv preprint arXiv:2402.01035*, 2024.
- [DSS_{24a}] F. Damoun, H. Seba, and R. State. Federated learning-based tokenizer for domain-specific language models in finance. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, 2024.
- [DSS_{24b}] F. Damoun, H. Seba, and R. State. Privacy-preserving behavioral anomaly detection in dynamic graphs for card transactions. In *International Conference on Web Information Systems Engineering*. Springer, 2024.
- [Eur₁₆] European Union. Regulation (EU) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (general data protection regulation), 2016. Accessed: 2024-08-27.
- [Eur₁₇] European Central Bank. The use of cash by households in the euro area. <https://www.ecb.europa.eu/pub/pdf/scpsps/ecbspl6.en.pdf>, 2017. ECB Occasional Paper Series No. 201.
- [Fed₁₉] Federal Trade Commission. Equifax to pay \$575 million as part of settlement with ftc, cfpb, and states related to 2017 data breach, 2019.

- [FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.
- [FLJ⁺14] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An {End-to-End} case study of personalized warfarin dosing. In *23rd USENIX security symposium (USENIX Security 14)*, pages 17–32, 2014.
- [FLL17] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1797–1806, 2017.
- [Flo62] Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962.
- [Fre79] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1:215–239, 1979.
- [GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.
- [GBD17] Joe Geumlek, Maria-Florina Balcan, and Christos Dimitrakakis. Rényi differential privacy. In *Advances in Neural Information Processing Systems*, pages 2600–2609, 2017.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 169–178. ACM, 2009.
- [GF18] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.
- [GJ19] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pages 2083–2092. PMLR, 2019.
- [GL16] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

- [GSR⁺17] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1263–1272. PMLR, 2017.
- [Har73] Frank Harary. *Graph Theory*. Addison-Wesley, 1973.
- [Haz22] Ahmed Hazourli. Financialbert-a pretrained language model for financial text mining. *Research Gate*, 2, 2022.
- [Hoc97] S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.
- [HT22] Yuxin He and Buzhou Tang. SetGNER: General named entity recognition as entity set generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2022.
- [HYC⁺22] Koki Horita, Bin Yang, Thomas Carette, Masanobu Jimbo, and Akihiro Nakao. Optimal network selection method using federated learning to achieve large-scale learning while preserving privacy. In *2022 IEEE 11th International Conference on Cloud Networking (CloudNet)*, pages 220–228. IEEE, 2022.
- [HYL17] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.
- [JVS⁺16] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.
- [KA24] Shima Khoshraftar and Aijun An. A survey on graph representation learning methods. *ACM Transactions on Intelligent Systems and Technology*, 15(1):1–55, 2024.
- [KCMW24] Asfia Kawnine, Hung Cao, Atah Nuh Mih, and Monica Wachowicz. Evaluating multi-global server architecture for federated learning. In *2024 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6. IEEE, 2024.
- [Kin14] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [KKYI23] Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Transformer language models handle word frequency in prediction head. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the*

- Association for Computational Linguistics: ACL 2023*, pages 4523–4535, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [KLN⁺11] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.
- [KMAS⁺19] Peter Kairouz, H Brendan McMahan, Maruan Al-Shedivat, Mehdi Bennis, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.
- [KMS⁺21] Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. Practical and private (deep) learning without sampling or shuffling. In *International Conference on Machine Learning*, pages 5213–5225. PMLR, 2021.
- [KOV15] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pages 1376–1385. PMLR, 2015.
- [KR18] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.
- [KRS⁺19] Anish Khazane, Jonathan Rider, Max Serpe, Antonia Gogoglou, Keegan Hines, C Bayan Bruss, and Richard Serpe. Deeptrax: Embedding graphs of financial transactions. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 126–133. IEEE, 2019.
- [KT19] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2, 2019.
- [KTA19] Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. *Advances in neural information processing systems*, 32, 2019.
- [KW17] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2017.
- [LBM23] Tomasz Limisiewicz, Jiří Balhar, and David Mareček. Tokenization impacts multilingual language modeling: Assessing vocabulary allocation and overlap across languages. *arXiv preprint arXiv:2305.17179*, 2023.

- [LBR₂₃] Iyadh Ben Cheikh Larbi, Aljoscha Burchardt, and Roland Roller. Clinical text anonymization, its influence on downstream nlp tasks and the risk of re-identification. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 105–111, 2023.
- [LCY⁺₁₈] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 2077–2085, 2018.
- [LDCH₂₂] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th international conference on data engineering (ICDE)*, pages 965–978. IEEE, 2022.
- [LE₂₄] Prabin B Lamichhane and William Eberle. Anomaly detection in graph structured data: A survey. *arXiv preprint arXiv:2405.06172*, 2024.
- [LHH⁺₂₁] Zhuang Liu, Degen Huang, Kaiyu Huang, Zhuang Li, and Jun Zhao. Finbert: A pre-trained financial language representation model for financial text mining. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*, pages 4513–4519, 2021.
- [LHZ⁺₂₁] Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Christophe Dupuy, Rahul Gupta, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. Fednlp: Benchmarking federated learning methods for natural language processing tasks. *arXiv preprint arXiv:2104.08815*, 2021.
- [LJZ⁺₂₁] Xiang Li, Meirui Jiang, Xiaoya Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021.
- [LLA⁺₂₃] Cedric Lothritz, Bertrand Leblot, Kevin Allix, Saad Ezzini, Tegawendé F Bissyandé, Jacques Klein, Andrey Boytsov, Clément Lefebvre, and Anne Goujon. Evaluating the impact of text de-identification on downstream nlp tasks. In *The 24rd Nordic Conference on Computational Linguistics*, 2023.
- [LLG⁺₁₉] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.
- [LLK₁₉] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International conference on machine learning*, pages 3734–3743. pmlr, 2019.

- [LMW⁺20] Xinyang Li, Xiaoqian Ma, Xiao Wang, Yuxin Ding, and Qian Zhang. Personalized privacy-preserving methods for centralized and federated learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1271–1280, 2020.
- [LSHL22] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70, 2022.
- [LSZ⁺20] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.
- [LTBZ16] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations*, 2016.
- [LWSV16] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *Advances in neural information processing systems*, pages 1885–1893, 2016.
- [LWY⁺19] Ziqi Liu, Dong Wang, Qianyu Yu, Zhiqiang Zhang, Yue Shen, Jian Ma, Wenliang Zhong, Jinjie Gu, Jun Zhou, Shuang Yang, et al. Graph representation learning for merchant incentive optimization in mobile payment marketing. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2577–2584, 2019.
- [LXW22] Pengrui Liu, Xiangrui Xu, and Wei Wang. Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives. *Cybersecurity*, 5(1):4, 2022.
- [LZL⁺22] Chuang Liu, Yibing Zhan, Chang Li, Bo Du, Jia Wu, Wenbin Hu, Tongliang Liu, and Dacheng Tao. Graph pooling for graph neural networks: Progress, challenges, and opportunities. *arXiv preprint arXiv:2204.07321*, 2022.
- [MAWY21] Guixiang Ma, Nesreen K Ahmed, Theodore L Willke, and Philip S Yu. Deep graph similarity learning: A survey. *Data Mining and Knowledge Discovery*, 35:688–725, 2021.
- [MBo8] S McAlearney and TJXD Breach. Ignore cost lessons and weep. *CIO*, page 565, August 7 2008.
- [MBM⁺17] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs

- and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5115–5124, 2017.
- [MCCD₁₃] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations (ICLR)*, 2013.
- [MCCJ₂₂] Casey Meehan, Amrita Roy Chowdhury, Kamalika Chaudhuri, and Somesh Jha. Privacy implications of shuffling. In *International Conference on Learning Representations*, 2022.
- [Mea₂₀] Gengchen Mai and et al. Multi-scale representation learning for spatial feature distributions using grid cells. In *International Conference on Learning Representations (ICLR)*, 2020.
- [Mem₂₄] Memgraph. Graph clustering algorithms: Usage and comparison. *Memgraph*, 2024. <https://memgraph.com/blog/graph-clustering-algorithms>.
- [MKB⁺₁₀] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Inter-speech*, volume 2, pages 1045–1048, 2010.
- [MMG₂₃] Simona Mazzarino, Andrea Minieri, and Luca Gilli. Nerpii: A python library to perform named entity recognition and generate personal identifiable information. In *Proceedings of the Seventh Workshop on Natural Language for Artificial Intelligence (NL4AI 2023) co-located with 22th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2023)*, 2023.
- [MMR⁺₁₇] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.
- [MRTZ₁₇] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.
- [MSDCS₁₉] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE, 2019.

- [MST₁₉] Daiki Matsunaga, Toyotaro Suzumura, and Toshihiro Takahashi. Exploring graph neural networks for stock market predictions with rolling window analysis. *arXiv preprint arXiv:1909.10660*, 2019.
- [MZM₂₃] Aabid A Mir, Megat F Zuhairi, and Shahrulniza Musa Musa. Graph anomaly detection with graph convolutional networks. *International Journal of Advanced Computer Science & Applications*, 14(11), 2023.
- [NCV⁺₁₇] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *ArXiv*, abs/1707.05005, 2017.
- [Newo₃] M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126, 2003.
- [NHW⁺_{11a}] EWT Ngai, Y Hu, YH Wong, Y Chen, and X Sun. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, 50(3):559–569, 2011.
- [NHW⁺_{11b}] EWT Ngai, Y Hu, YH Wong, Y Chen, and X Sun. The application of data mining techniques in financial fraud detection: a classification framework and an academic review of literature. *Decision Support Systems*, 50(3):559–569, 2011.
- [NJL₂₁] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*, 2021.
- [NTPV₂₀] Anmol Nayak, Hariprasad Timmapathini, Karthikeyan Ponnalagu, and Vijendran Gopalan Venkoparao. Domain adaptation challenges of bert in tokenization and sub-word representations of out-of-vocabulary words. In *Proceedings of the first workshop on insights from negative results in NLP*, pages 1–5, 2020.
- [PARS₁₄] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.
- [PCWF₀₇] Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 201–210, 2007.

- [PEV₁₉] Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. Bpe-dropout: Simple and effective subword regularization. *arXiv preprint arXiv:1910.13267*, 2019.
- [PL₂₃] Constantinos Patsakis and Nikolaos Lykousas. Man vs the machine in the struggle for effective text anonymisation in the age of large language models. *Scientific Reports*, 13(1):16026, 2023.
- [PLMTB₂₄] Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. Language model tokenizers introduce unfairness between languages. *Advances in Neural Information Processing Systems*, 36, 2024.
- [Prz₀₇] Nataša Przulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.
- [RNSS₁₈] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. In *OpenAI Research Covers.*, 2018.
- [RPV⁺₂₁] Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135. Association for Computational Linguistics, August 2021.
- [RSKH₁₅] Stephen Ranshous, Sihang Shen, Danai Koutra, and Sarah Harenberg. Anomaly detection in dynamic networks: a survey. *Computational Statistics*, 7(3):223–247, 2015.
- [RSR⁺₂₀] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.
- [RST₂₀] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5470–5477, 2020.
- [RWC⁺₁₉] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

- [RZH⁺20] Susie Xi Rao, Shuai Zhang, Zhichao Han, Zitao Zhang, Wei Min, Zhiyao Chen, Yinan Shan, Yang Zhao, and Ce Zhang. xfraud: explainable fraud transaction detection. *arXiv preprint arXiv:2011.12193*, 2020.
- [RZZ⁺21] Yuxiang Ren, Hao Zhu, Jiawei Zhang, Peng Dai, and Liefeng Bo. Ensemfdet: An ensemble approach to fraud detection based on bipartite graph. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2039–2044. IEEE, 2021.
- [SBN⁺21] Susheel Suresh, Vinith Budde, Jennifer Neville, Pan Li, and Jianzhu Ma. Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1541–1551, 2021.
- [SCE⁺22] Raj Sanjay Shah, Kunal Chawla, Dheeraj Eidnani, Agam Shah, Wendi Du, Sudheer Chava, Natraj Raman, Charese Smiley, Jiaao Chen, and Diyi Yang. When flue meets flang: Benchmarks and large pretrained language model for financial domain. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2022.
- [Sha51] Claude E Shannon. Prediction and entropy of printed english. *Bell system technical journal*, 30(1):50–64, 1951.
- [SHB15] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.
- [SHH⁺18] Lichao Sun, Lifang He, Zhipeng Huang, Bokai Cao, Congying Xia, Xiaokai Wei, and S Yu Philip. Joint embedding of meta-path and meta-graph for heterogeneous information networks. In *2018 IEEE international conference on big knowledge (ICBK)*, pages 131–138. IEEE, 2018.
- [SKSM08] Amlan Srivastava, Amlan Kundu, Shamik Sural, and Arun K Majumdar. Credit card fraud detection using hidden markov model. *IEEE Transactions on Dependable and Secure Computing*, 5(1):37–48, 2008.
- [SR23] Ofir Ben Shoham and Nadav Rappoport. Federated learning of medical concepts embedding using behrt. *arXiv preprint arXiv:2305.13052*, 2023.
- [SRB07] Gábor J Székely, Maria L Rizzo, and Nail K Bakirov. Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35:2769–2794, 2007.

- [SS98] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. *technical report, SRI International*, 1998.
- [SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.
- [SSVL⁺11] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.
- [SVP⁺09] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pages 488–495. PMLR, 2009.
- [Swe02] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems*, 10(05):557–570, 2002.
- [SWS⁺23] Xiaona Song, Nana Wu, Shuai Song, Yijun Zhang, and Vladimir Stojanovic. Bipartite synchronization for cooperative-competitive neural networks with reaction–diffusion terms via dual event-triggered mechanism. *Neurocomputing*, 550:126498, 2023.
- [SWSS23] Xiaona Song, Nana Wu, Shuai Song, and Vladimir Stojanovic. Switching-like event-triggered state estimation for reaction–diffusion neural networks against dos attacks. *Neural Processing Letters*, 55(7):8997–9018, 2023.
- [SX18] Lili Su and Jiaming Xu. Securing distributed machine learning in high dimensions. *arXiv preprint arXiv:1804.10140*, 2018.
- [SZK⁺22] Toyotaro Suzumura, Yi Zhou, Ryo Kawahara, Nathalie Baracaldo, and Heiko Ludwig. Federated learning for collaborative financial crimes detection. In *Federated learning: A comprehensive overview of methods and applications*, pages 455–466. Springer, 2022.
- [Tar72] Robert E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.
- [TL11] Hanghang Tong and Ching-Yung Lin. Non-negative residual matrix factorization with application to graph anomaly detection. In *Proc. SIAM Int. Conf. Data Mining*, pages 1–11, 2011.

- [TLI⁺₂₃] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [TQW⁺₁₅] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077, 2015.
- [TWL⁺₂₂] Yuanyishu Tian, Yao Wan, Lingjuan Lyu, Dezhong Yao, Hai Jin, and Lichao Sun. Fedbert: When federated learning meets pre-training. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–26, 2022.
- [TYŞO_{23a}] Cagri Toraman, Eyup Halit Yilmaz, Furkan Şahinuç, and Oguzhan Ozelik. Impact of tokenization on language models: An analysis for turkish. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(4):1–21, 2023.
- [TYŞO_{23b}] Cagri Toraman, Eyup Halit Yilmaz, Furkan Şahinuç, and Oguzhan Ozelik. Impact of tokenization on language models: An analysis for turkish. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(4):1–21, 2023.
- [VBDV₂₃] Mark Vero, Mislav Balunović, Dimitar Iliev Dimitrov, and Martin Vechev. Tableak: Tabular data leakage in federated learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 35051–35083. PMLR, 2023.
- [VCC⁺₁₇] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [VCC⁺₁₈] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.
- [VSKB₁₀] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010.
- [VSP⁺₁₇] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

- [WBS⁺22] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and S Yu Philip. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Transactions on Big Data*, 9(2):415–436, 2022.
- [WCG20] Changhan Wang, Kyunghyun Cho, and Jiatao Gu. Neural machine translation with byte-level subwords. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9154–9160, 2020.
- [WDL⁺19] Yueyang Wang, Ziheng Duan, Binbing Liao, Fei Wu, and Yueting Zhuang. Heterogeneous attributed network embedding with graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 10061–10062, 2019.
- [Wes01] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2nd edition, 2001.
- [WHJ⁺09] Christopher Whitrow, David J Hand, Paul Juszczak, David J Weston, and Neil M Adams. Transaction aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery*, 18(1):30–55, 2009.
- [WL21] Wenqi Wei and Ling Liu. Gradient leakage attack resilient deep learning. *IEEE Transactions on Information Forensics and Security*, 17:303–316, 2021.
- [WLLJ20] Jinhyun Wang, Ping Liang, Guangyu Liu, and Gauri Joshi. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *NeurIPS*, 2020.
- [WLQ⁺22] Yanbo J Wang, Yuming Li, Hui Qin, Yuhang Guan, and Sheng Chen. A novel deberta-based model for financial question answering task. *arXiv preprint arXiv:2207.05875*, 2022.
- [WLW⁺23] Haifeng Wang, Jiwei Li, Hua Wu, Eduard Hovy, and Yu Sun. Pre-trained language models and their applications. *Engineering*, 25:51–65, 2023.
- [WPC⁺20] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.
- [WS98] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393:440–442, 1998.
- [WSZ⁺19] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pages 6861–6871. PMLR, 2019.

- [Wu₂₁] Xianchao Wu. Finmegatron: Large financial domain language models. *Proceedings of the Second Type Research Meeting*, 2021(FIN-026):22, 2021.
- [WYS⁺₂₀] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.
- [WZL⁺₁₆] Faqiang Wang, Wangmeng Zuo, Liang Lin, David Zhang, and Lei Zhang. Joint learning of single-image and cross-image representations for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1288–1296, 2016.
- [WZT⁺₂₃] Rui Wang, Zhihe Zhuang, Hongfeng Tao, Wojciech Paszke, and Vladimir Stojanovic. Q-learning based fault estimation and fault tolerant iterative learning control for mimo systems. *ISA transactions*, 142:123–135, 2023.
- [WZXS₂₂] Jianian Wang, Sheng Zhang, Yanghua Xiao, and Rui Song. A review on graph neural network methods in financial applications. *Journal of Data Science*, 20(2):111–134, 2022.
- [XHCL₁₉] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. DbA: Distributed back-door attacks against federated learning. In *International Conference on Learning Representations*, 2019.
- [XHLJ₁₈] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.
- [YCA⁺₁₈] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2672–2681, 2018.
- [YLCT₁₉] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.
- [YLW₂₃] Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. Fingpt: Open-source financial large language models. *FinLLM Symposium at IJCAI 2023*, 2023.
- [YMV⁺₂₁] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16337–16346, 2021.

- [YV15a] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.
- [YV15b] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM, 2015.
- [YWCW19] Yiyang Yang, Zhongyu Wei, Qin Chen, and Libo Wu. Using external knowledge for financial event prediction based on graph neural networks. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 2161–2164, 2019.
- [YXZ⁺20] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. Heterogeneous network representation learning: A unified framework with survey and benchmark. *IEEE Transactions on Knowledge and Data Engineering*, 34(10):4854–4873, 2020.
- [YYM⁺18] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.
- [YZCL19] Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. Neural network inversion in adversarial setting via background knowledge alignment. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 225–240, 2019.
- [YZY⁺19] Wensi Yang, Yuhang Zhang, Kejiang Ye, Li Li, and Cheng-Zhong Xu. Ffd: A federated learning based method for credit card fraud detection. In *Big Data–BigData 2019: 8th International Congress, Held as Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, June 25–30, 2019, Proceedings 8*, pages 18–32. Springer, 2019.
- [Zea20] EdD. Zhong and et al. Reconstructing continuous distributions of 3d protein structure from cryo-em images. In *International Conference on Learning Representations (ICLR)*, 2020.
- [ZJP⁺20] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 253–261, 2020.
- [ZKM⁺20] Wennan Zhu, Peter Kairouz, Brendan McMahan, Haicheng Sun, and Wei Li. Federated heavy hitters discovery with differential privacy. In *International Conference on Artificial Intelligence and Statistics*, pages 3837–3847. PMLR, 2020.

- [ZLH19] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.
- [ZLWP21] Yizhen Zheng, Vincent CS Lee, Zonghan Wu, and Shirui Pan. Heterogeneous graph attention network for small and medium-sized enterprises bankruptcy prediction. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 140–151. Springer, 2021.
- [ZMB20] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.
- [ZMG⁺23] Vilém Zouhar, Clara Meister, Juan Luis Gastaldi, Li Du, Tim Vieira, Mrinmaya Sachan, and Ryan Cotterell. A formal perspective on byte-pair encoding. *arXiv preprint arXiv:2306.16837*, 2023.
- [ZSX⁺18] Jiaxuan Zhang, Xiaokai Shi, Junyuan Xie, Hongyuan Ma, and Irwin King. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.
- [ZYGW21] Wenbo Zheng, Lan Yan, Chao Gou, and Fei-Yue Wang. Federated meta-learning for fraudulent credit card detection. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4654–4660, 2021.
- [ZZL15] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

