CaPE: Category Preserving Embeddings for Similarity-Search in Financial Graphs

Gaurav Oberoi
Mastercard
Gurugram, India
gaurav.oberoi@mastercard.com

Pranav Poduval
Mastercard
Gurugram, India
pranav.poduval@mastercard.com

Karamjit Singh
Mastercard
Gurugram, India
karamjit.singh@mastercard.com

Sangam Verma
Mastercard
Gurugram, India

Pranay Gupta
Mastercard
Gurugram, India

ABSTRACT

Similarity-search is an important problem to solve for the payment industry having user-merchant interaction data. It finds out merchants similar to a given merchant and solves various tasks like peer-set generation, recommendation, community detection, and anomaly detection. Recent works have shown that by leveraging interaction data, Graph Neural Networks (GNNs) can be used to generate node embeddings for entities like a merchant, which can be further used for such similarity-search tasks. However, most of the real-world financial data come with high cardinality categorical features such as city, industry, super-industries, etc. which are fed to the GNNs in a one-hot encoded manner. Current GNN algorithms are not designed to work for such sparse features which makes it difficult for them to learn these sparse features preserving embeddings. In this work, we propose CaPE, a Category Preserving Embedding generation method which preserves the high cardinality feature information in the embeddings. We have designed CaPE to preserve other important numerical feature information as well. We compare CaPE with the latest GNN algorithms for embedding generation methods to showcase its superiority in peer set generation tasks on real-world datasets, both external as well as internal (synthetically generated). We also compared our method for a downstream task like link prediction.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; *Semi-supervised learning settings.*

KEYWORDS

Graph Neural Networks, Financial Graphs, Similarity Search, Embeddings, Sparse Features

ACM Reference Format:
Gaurav Oberoi, Pranav Poduval, Karamjit Singh, Sangam Verma, and Pranay Gupta. 2022. CaPE: Category Preserving Embeddings for Similarity-Search in Financial Graphs. In *3rd ACM International Conference on AI in Finance (ICAIF '22), November 2–4, 2022, New York, NY, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3533271.3561788

ICAIF '22, November 2–4, 2022, New York, NY, USA

ACM ISBN 978-1-4503-9376-8/22/11…$15.00
https://doi.org/10.1145/3533271.3561788

1 INTRODUCTION

The payment industry consists of a huge data pertaining to the interaction of users with a merchant, where a user makes a transaction at a merchant through various channels like offline-Point of Sale (POS), an online gateway, or a contactless mechanism. A user might interact with hundreds to thousands of merchants over its life cycle and these interactions could signify different things like the user's preference for a certain merchant or a product, or the merchant's popularity among a certain set of users. Companies make use of this data for a variety of different tasks like, a) generating performance reports for the merchants in terms of average sales volume, fraud rates, and decline rates as compared to the merchants where a similar set of users interacted, b) recommending merchants to a user based on its past interactions, c) performing clustering analysis to detect communities of merchants, or finding out the anomalous merchants [14]. These tasks help in increasing the merchant's sales, enhancing the product's visibility, or anticipating a fraud thereby reducing inefficiencies and increasing the company's revenue.

All these tasks require the companies to search for the entities (here merchants) similar to a given entity (merchant). The heart of the solution lies in finding a good representation for the merchant set based on some similarity score. A naive way is to run clustering such K-means [21] and k-nearest Neighbour algorithms [2] over the raw feature set, however these algorithms fail to provide good results for a high-dimensional feature set which is distributed non-linearly over the feature-space. To solve this problem, data transformation schemes like PCA [23] and Auto-Encoders (AE) [20] have been extensively utilized in the last decade, but the latest research suggests that there is an appreciable amount of information loss, and we can do much better.

By utilizing user-merchant interactions, we can better represent a merchant in accordance with the user preferences derived from merchant's provided services. Merchants that are linked to a specific set of users must have some common attributes as they are desired by the same set of users. For example, users who like to have Japanese cuisines would be transacting at a lot of Japanese restaurants. This information is very difficult to capture from

transaction features alone. So, to capture such information, user-merchant interaction can be modelled as a Bipartite graph with users and merchants as two types of nodes and edges represented by the transactions between them. As shown in Figure 1, bipartite graphs can also be converted to homogeneous graphs which have only one type of nodes. An edge in a homogeneous merchant graph may represent connectivity via minimum number of common users. By utilizing such a graph structure, a merchant can be represented as a function of its interactions with users and other merchants.

In recent years, Graph Neural Networks (GNNs) have emerged out to be a powerful tool in harnessing interactive data like user-merchant data for generating representations for the nodes. Graph Convolution Networks (GCNs) [15], Graph Sample and Aggregate (GraphSAGE) [10] and GAT [27] have suggested a way to generate the node embeddings in transductive (non-generalizable) and inductive (generalizable) settings respectively. These works aggregate the neighborhood node's features along with the node's self features to generate the embeddings for a given node. They take into account both graph structure information and node feature information in generating the embedding representation. Recently, PanRep [12] suggested a way to generate universal embedding representation which serves multiple tasks by incorporating multiple loss functions.

In a real-world financial graph, a lot of categorical information exists which can be used to describe the node's properties or features. For example, for a merchant node in a user-merchant interaction graph, features like merchant category, industry type, merchant zip, city, etc are used to characterize a merchant node. These categorical features can be of a high cardinality, and are fed to the GNN algorithm using one-hot encoding, which makes the graph's node features highly sparse. A limitation of current GNN algorithms is that they use neighborhood aggregation over these features for generating a node's representation, which is not designed to work for such sparse features [4]. Most of the category information which is required by the downstream application is not preserved in these embeddings due to the curse of dimensionality. Also, we may want to selectively give more importance to some numerical features over the others, for example in the user-merchant graph, we want GNN to bring merchants with higher physical proximity, closer to each other in the embedding space. This can be done by giving more importance to latitude-longitude features.

In this work, we propose CaPE, a framework to generate category-preserving embeddings which also preserved important numerical feature information. The embeddings thus generated lead to better similarity-search and improved performance on downstream tasks. We propose to use two GNN modules:

- Module-1 trained in a semi-supervised manner by using high-cardinality categories as labels, allowing us to use the remaining features and neighbouring information to learn about these sparse features. For an already existing node, our framework generates much superior embeddings, as compared to vanilla GNN algorithms. For a new node, since most of the information is preserved in the graph structure, sparse feature information is not required to generating a high-quality embedding. Also, to solve for selective weighing, we propose to use a weighted unsupervised loss function along with the supervised loss, where the edge weight between two nodes is derived from important numerical features. In the homogeneous merchant-merchant graph, the edge weights can be derived using the merchant's latitude and longitude features to capture the merchant's distance information. In this way, the important numerical information is also preserved along with the categorical information, generating superior embeddings.
- Module 2 is trained in an unsupervised setting using only high-cardinality categories as input features and optimizing GNN algorithm only on them. This GNN module is used to capture any complementary category information which is left by the first module such as higher-order interactions among the categories.
- We combine the output embeddings of a Module 1 and Module 2 through a self-attention mechanism to get our final embeddings which is then trained via an unsupervised loss function

To adhere to data privacy regulations, we show our results on an anonymous graph structure with synthetically generated features having statistical similarity to the real data. We also validate our results on open-source Elo Merchant data available on Kaggle.[1] The main contributions of this work are summarized below:

- We solve merchant similarity problem using graph learning based on a homogeneous merchant graph.
- We present CaPE, a novel algorithm to generate node embeddings that preserve high-cardinality sparse category information as well as retaining important numerical feature information
- We introduce novel notion of similarity for merchants, by proposing 4 similarity metrics that are highly relevant to the Payment Industry
- We conduct comprehensive experiments on real-world datasets both internal as well as external to demonstrate the effectiveness of the proposed framework.

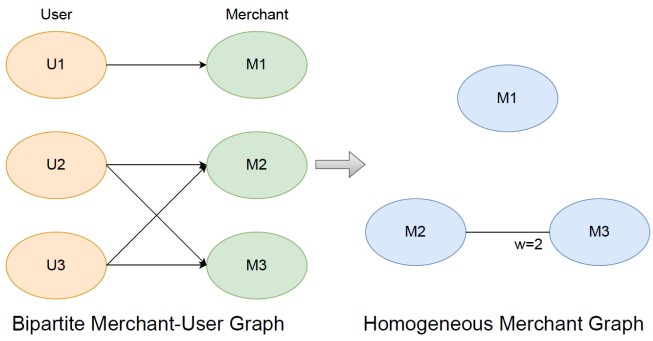

Figure 1: Graphs in Payment Industry. Converting Bipartite User-Merchant Graphs to a Homogeneous Merchant-Merchant Graph

[1] https://www.kaggle.com/competitions/elo-merchant-category-recommendation/data

2 RELATED WORK

Classical approaches over graphs such as spectral clustering, PageRank algorithm, etc have inspired a lot of current embedding generation techniques. These techniques such as Node2vec [9], Deepwalk [24], Line [26] learn embeddings using a random walk and a matrix factorization based loss function which don't allow them to generalize over new graph. Owing to the success of convolutional neural networks (CNNs) in structured data modelling, a lot of CNN approaches like [6], [3], [16], [22] have also been proposed in the field of graph learning. These approaches have shown promising results but are non-generalizable to newer nodes and are non-scalable for larger graphs.

Recent works have proposed inductive and hence generalizable approaches like Graph Attention Networks (GAT) [27], GraphSAGE [10] and variants [25], [17] for homogeneous graph embedding generation. These works do not compromise in accuracy when compared to GCN, are generalizable to new nodes and the weights learned can also be used for transfer learning in an altogether different graph. We have used the architecture proposed by GraphSAGE for our work, because of its robust performance across different tasks and ease of implementation.

A lot of modifications have also been made to these approaches to generate embeddings for bipartite graphs in works such as Metapath2Vec [5], BiNE [7], CascadeBGNN [11]. Bipartite graph approaches are computationally expensive in nature as we have order of magnitude more user nodes when compared with merchant nodes. The dataset is sparse, heterogeneous, and billion scale imposing a challenge towards both the algorithmic side and the implementation side. Therefore we resort to converting our graph structure to a homogeneous one (homogeneous merchant graph) and using homogeneous graph GNN algorithm over it.

Also, recently proposed recommendation systems [8], [18], [28], [29] have taken graph embedding generation route for making product recommendation. These works have applied graph embedding algorithms on millions of entities such as user, product for the industry application. Some works have also been proposed in the financial space like DeepTrax [14] which have used representation learning on credit card data. These works in financial space utilize merchant-user bipertite graph for generating embeddings, however as discussed above representation learning on a bipertite graphs is a challenging problem to solve.

In 2020, PanRep, a work for generating universal node embeddings in heterogeneous graphs was proposed. PanRep uses multiple decoder paths and multiple loss functions over a bipartite graph to generate universal embeddings. Inspired by PanRep's usage of multiple loss functions and to address the problems of sparsity associated with node features and heterogeneity associated with bipartite graph, we propose CaPE, a framework for generating category preserving Merchant embeddings using spare-feature based loss function applied over a homogeneous graph. We also tweak the upsupervised loss function to selectively give more importance to some numerical features over the others.

3 DATASETS

3.1 Internal Synthetic Data

Obliged by our internal controls to data privacy and protection, we do not use real transaction data features for any merchant or user. We first generate a merchant graph containing information only about whether the two merchants are connected via a user during some timeline T or not. After generating graph, we anonymize merchant identities along with the industry label information of the merchants. Then, for the merchants belonging to each of the industries, we separately generate synthetic merchant features using SMOTE-NC technique on normalized real transaction data [1]. The features thus generated are normalized and statistically similar to normalized real data, but original transaction level or merchant aggregate level information cannot be traced back using them. We use six month's merchant-user interactions to generate the graph structure. We then remove merchants with extremely low and high transactions during the 6 months as these merchants will add noise and bias in the graph respectively. The graph contains 212k unique merchants and and 41M edges between them. We anonymize the graph keys so that no merchant inference can be made using it.

Merchants Features: We use average ticket price, total sales volume, average spent per card, for last 1 week, 1 month and 3 months as the merchant features. We also use latitude, longitude, industry, and super-industry information along with it.

Synthetic Data Generation: For merchants belonging each of the 100 anonymized industries, we separately find out the synthetically generated features representing average ticket price, total sales volume, average spent per card, latitude and longitude using SMOTE-NC. We generate synthetic dataset at industry level instead of the complete dataset because the distribution of data for each of the industries is different and using SMOTE on complete data would completely loose industry level insights from the features.

3.2 Elo Merchant Data

We also validate our results on open-source, Elo Merchant Category Recommendation data available at Kaggle. It consists of transaction level information of up to 3 months' worth of transactions for every card at any of the provided merchants. Two separate transaction level files are provided for traning and testing, each consisting of overall 12 months of transaction data. It also contains one separate merchant file containing information about the merchant category and sales/purchase patters like overall-category, sub-sector, city, state, average sales/purchase with lag of 3/6/12 months.

We have sampled 6 months of transaction from training file and used them to generate the Merchant graph structure. Similar to our internal data, we have removed merchants with very low and very high transaction volume, to remove noise from the graph. The graph thus generated consists of 1.5M edges and 98k unique merchants.

4 METHODOLOGY

In this section we describe the working of CaPE in detail. We first describe the transformation of a transaction bipartite graph to a homogeneous graph, then give the overview of GNN algorithm and finally discuss our strategy for generating optimal embeddings for

the task of similarity-search. Figure 2 shows the overview of our methodology.

4.1 Homogeneous Merchant Graph

We select user-merchant interactions for a specific timeline T to generate a bipartite graph $G_b = (M, U, E)$ with M, U representing the sets of merchants, users interacting in the timeline and E representing the edges or interactions between them. For a merchant m_i and user u_j in the graph G_b, there exists an edge e_{ij} if user u_j transacted with merchant m_i during the timeline T.

We transform G_b to a homogeneous merchant graph $G = (M, E')$ from G_b with M being set of Merchants active during timeline T and E' being a set of edges we design to achieve this transformation. For the merchants m_i and m_j in the graph G, there exists an edge e'_{ij} if these are connected via more than 1 ($M - U - M$) path in the bipartite graph G_b. In other words, edge e'_{ij} will exist if there are more than 1 users u connecting merchants m_i and merchant m_j during the timeline T.

Another way to transform this is to create a weighted homogeneous Graph G_w with w being the number of $M - U - M$ paths (number of common users) and use it to generate embedding. However, since majority of graph learning work is in unweighted space, we use a simplified unweighted graph for our application. Another way could be to increase the threshold of minimum number of $M - U - M$ paths required in G_b for an edge to exists between the two merchants in graph G. This can further help reduce the noise in the Graph G and hence the final embeddings. We use at least 2 common paths as it's inline with our business requirements.

4.2 GNN Algorithm

We use inductive neighborhood aggregation strategy defined in GraphSAGE [10] for generating the merchant embeddings. This strategy is based on the assumption that densely connected nodes in the graph are similar to each other and should have similar representations in the embedding space.

For the k^t iteration step, for each merchant $m \in M$, we first aggregate the representations of the neighbours $\mathcal{N}_m$ as $h^k_{\mathcal{N}(m)} = \mathcal{A}(h^{k-1}_u, \forall u \in \mathcal{N}(m))$ where h^{k-1}_u is the representation of node u from the $(k-1)^{th}$ step. $\mathcal{A}$ is an aggregator function which aggregates information a merchants' local neighborhood. We concatenate neighbourhood aggregation of merchant m, $h^k_{\mathcal{N}(m)}$ with the merchant representation form the previous step h^{k-1}_m and then feed this concatenated vector into a fully connected layer with a non-linear activation function σ. Finally, we normalize it to get the merchant representation for the current step.

An aggregator, $\mathcal{A}$ is required to have good representation capacity and symmetric property allowing aggregation arbitrarily over neighbourhood merchants. Mean, LSTM, Max Pooling, Mean Pooling are the commonly used ones having these properties. For this work as well, we use Mean-Pooling aggregator owing to its high representation capacity and low aggregation time complexity. It first feeds a merchant's neighbourhood embedding vectors into a fully-connected neural network and then use a mean-pooling operator over them for information aggregation.

$$\mathcal{A}^{pool}_k = mean(\{\sigma(W_{pool} z^k_{m_i} + b), \forall m_i \in \mathcal{N}(m)\}) \quad (1)$$

Here *mean* is element wise max operation and σ is a ELU activation function.

4.3 Category Preserving Embeddings

An unsupervised GNN algorithm is designed to preserve local neighbourhood structural information and the nodes' feature information. However high-cardinailty category information which is fed in the form of sparse one-hot encoded features remain partially preserved due the the curse of dimensionality [4], [19]. We propose to preserve this information by utilizing two different GNN modules trained differently using different settings of loss functions and input features.

4.3.1 GNN1: Loss-Based Category Preservation: We shift the sparse categorical features from input to the output space in the form of labels and then train a GNN module by adding supervised loss function based on the labels added. This allows the direct flow of gradient towards optimizing on the categorical features thereby preserving category information. Since this category information is correlated with other features, this loss function also helps in generating an overall better quality embeddings.

Category information using the above supervised loss is even preserved for any new node for which embedding is required to be generated in the inductive settings. This is because the new node is not an *iid* sample, but is connected to the correlated nodes in the graph structure and optimization over the correlated nodes preserves the category information for the new node as well, even without explicitly passing category information for the new node.

To selectively give more importance some numerical features over the others, based on the given application, we use a weighted unsupervised loss. Here we chose weights for a given edge by using the L2 distance between the important features of source and destination nodes. Node pairs (edges) with higher L-2 distance are given more weightage in the unsupervised loss function. This preserves the important numerical feature information in the embeddings.

Supervised loss: To preserve category information for a given merchant m, we use supervised multi-class cross entropy loss defined below:

$$L_{sup}(z_m, y) = -\sum_{l \in L} y_l \; log(p(\hat{y}(z_m) = l)) \quad (2)$$

Where l is label for merchant m. Our experimentation shows considerable improvement in embedding quality when using the above features as labels in the loss function.

Unsupervised Loss: The graph context loss function used in the previous works brings connected nodes closer to each other in the embedding space, while enforcing disparate nodes to have highly distinct embeddings. The unsupervised loss function is defined below:

$$L(z_m) = -log(\sigma({z_m}^T z_v)) \; - \; N.\mathbb{E}_{v_n \sim P_n(v)} \; log(\sigma(-{z_m}^T z_{v_n})) \quad (3)$$

where z_m is the embedding representation of merchant m, v is a merchant that occurs in the neighborhood of m on a fixed length random walk, σ is the sigmoid function, P_n is the negative sampling distribution, N is the number of negative samples and z_{v_n} is the embedding representation of the negative samples extracted from the distribution. The node embeddings used here are generated by aggregating the neighborhood node embeddings as discussed above

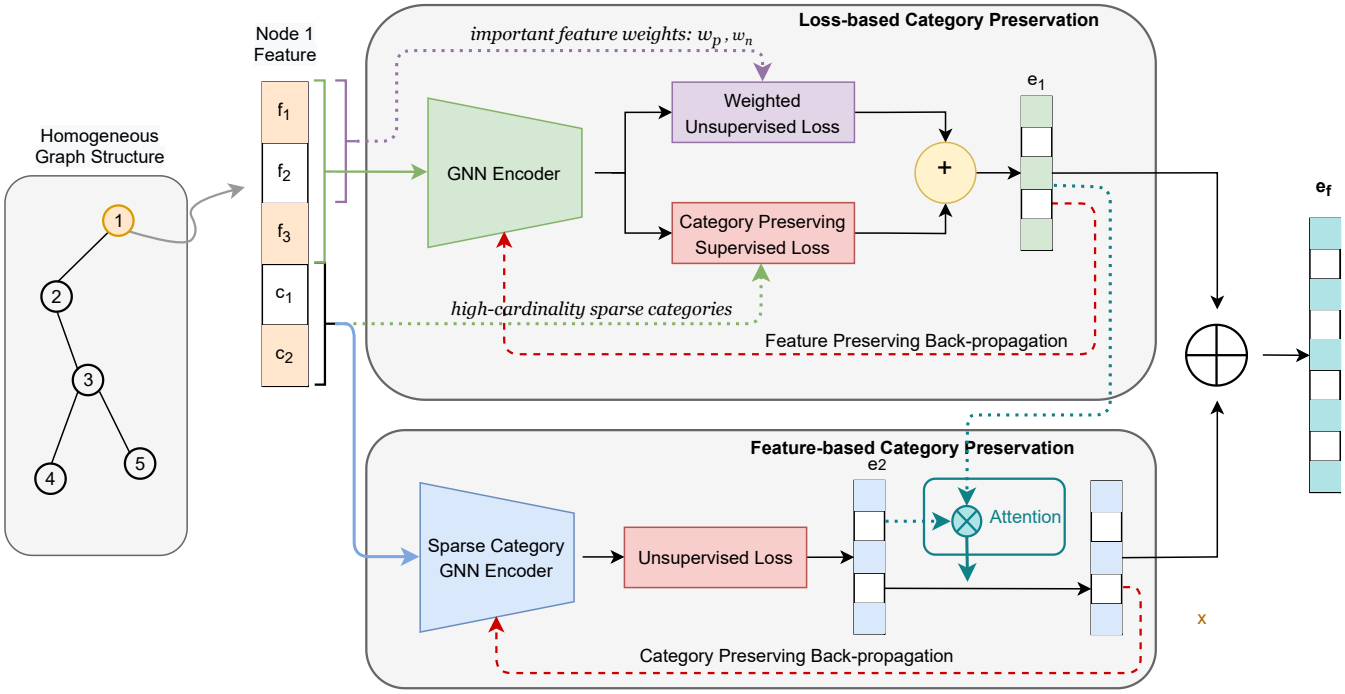

Figure 2: Overview of CaPE algorithm

and in detail in [10]. This loss function brings connected merchants closer to one another in the embedding space, which means that by designing a homogeneous graph, merchants with user proximity should have similar embeddings. However, in industrial applications, there may be proximities based on other features which need to be preserved. For example, for our internal dataset, we try to capture physical proximity based on the L2 distance between the merchants' lat-long features. We modify the above loss function to give more weightage to merchants with higher physical proximity. We modify the above loss function to give more weightage to some specific pair of merchants as compared to others.

$$L(z_m) = -w_p \; log(\sigma({z_m}^T z_v)) - w_n \; N.\mathbb{E}_{v_n \sim P_n(v)} \; log(\sigma(-{z_m}^T z_{v_n})) \quad (4)$$

$$w_p = \frac{1}{\sigma(||f_m - f_v||^2)} \quad (5)$$

$$w_n = \sigma(||f_m - f_{v_n}||^2) \quad (6)$$

Here $f_m \in \mathbb{F}_i$ represents the feature vector corresponding to the selected important numerical features for the merchant m. w_p and w_n enforce that merchants with proximity in the feature space $\mathbb{F}_i$ have similar embeddings while merchants which are distant in $\mathbb{F}_i$ have distinct embeddings.

4.3.2 GNN2: Feature-Based Category Preservation. We solved for the problem of optimization over sparse features using the above discussed GNN module, however there might be some information in the categories which is not learned using the supervised loss function. This complementary information required complex interactions among the features corresponding to categorical variables. To capture such complex information we shift back categories to the input as one-hot variables and train a GNN using only the sparse-category input features and the unsupervised loss defined in Eq. 3.

4.4 Merchant Similarity Search

Given a query vector $x \in \mathbb{R}^n$ and a set of vectors $[y_i]$, the problem of finding similar entities boils down to finding k nearest neighbors of x in terms of euclidean distance.

$$L = \arg\min_i \; ||x - y_i||_2 \quad (7)$$

where x and y_i are the embedding vectors generated via above discussed method. We use the euclidean distance in the embedding space to find out k nearest merchants for a given merchant. This search operation has a complexity of $O(n)$. The implementation for billion scale merchant search [13] is left as a future work.

5 EXPERIMENTS

To evaluate the quality of embeddings generated and quantity of information preserved by our method, we perform merchant similarity analysis as well as use the embeddings for a downstream

link prediction task. We also use t-SNE visualizations to show the goodness of our method in the generating the embeddings.

We examine the efficacy of our algorithm and its components, by benchmarking it's performance against the state-of-the-art graph node's embedding generation algorithms for both bipartite and homogeneous graphs namely BiGI, GraphSAGE and GAT. For our experiments we have chosen GraphSAGE as our base GNN algorithm because a) it's feature aggregation based b) its inductive. As shown in Table 4, we choose ELU non-linearity, aggregation depth of $K = 2$ with neighbourhood samples of $S1 = 25$ and $S2 = 10$. For all our experiments, we train the models using ADAM optimizer with a learning rate of $1e^{-4}$ for 2 epochs, keeping 5 negative samples with a batch size of 512. We performed our experiments on two datasets: internal as well as external, as shown in Table 3

5.1 Merchant Similarity Analysis

Notion of Similarity: The notion of similarity can vary depending on the application. The definition given below are derived from business understanding and we believe they serve a general purpose of finding similar merchants given a merchant across domains.

- **User Proximity**: We define this as the number of common users between two merchants. The more the number of common users between them, the higher the similarity between the two merchants. This serves as an indicator of the similarity of the merchants in terms of buyer preferences, for example, a user who prefers watching movies for recreation regularly at theatres might be visiting multiple theatres in that region, but might not be visiting any gaming zone or a club although all of these comes under entertainment industry. There is a high chance that this user eats more frequently at theatre food counters as compared to normal restaurants. Hence, the cinema and its food counter serve the same user base and are similar in terms of user preference.
- **Category Proximity**: For two merchants to be similar, they should belong the same category defining its broader class depending upon the application. This enables a fair comparison as merchant belonging to different categories can have different economic behaviour.
- **Important Feature Proximity**: For the two merchants to be considered similar and recommended against each other, they should be nearby in the euclidean space of the important feature. For our internal data, we used lat-log as the important feature so that merchants with physical proximity are suggested and a user can easily visit the similar merchant, if the merchant serves the user's requirement.
- **Sales Volume**: Sales volume is the total sales done by a merchant in a fixed time window. This is a major economic activity indicator and covers various dimensions of a merchants attributes like merchant category, sub-category, services provided, user base etc.

We design our experiment by randomly selecting 1000 merchants as benchmark merchants and find out their 10 nearest neighbours (*called peer-set*) using the embeddings generated. Aligned with the notion of similarity, we use the below four metrics to investigate the performance of these algorithms:

- Common Users: Average number of common users between the benchmark merchants and their peer set.
- Category Preserved: Percentage of merchants in the peer set having same category as benchmark merchant.
- Important Feature Deviation: Percentage deviation or L2 distance between the important feature of benchmark merchants and their peer set.
- Spend Deviation: Percentage deviation in average ticket price of the benchmark merchants from their peer set.

Table 1 shows a comparative study of the model performances. We can clearly see in our Internal as well on Elo's merchant Data, CaPE has embeddings where the merchants similar in terms of the important features and similar categories are brought closer together.

For the Internal Data, CaPE with category loss function produces a better peer set than the Homogeneous GraphSAGE, in terms of number of common users with an approximate 6-9% improvement. CaPE also has the 20% relative reduction in spending deviation implying merchants with similar spending patterns have been successfully clustered together. On the internal data CaPE also has 2-10% improvement in terms of Industry category preserved.

CaPE provides a significant improvement in the performance in all the 4 metrics even on the external Evo's merchant dataset. The peer set generated by CaPE is superior by 30% in terms of preserving the category. Comparing the numerical features we can see a significant improvement of 200%, while also having a 4-15% reduction in the spending deviation. The performance of CaPE on all these metrics suggests that CaPE produces representations for merchants that preserve the categorical and numerical features, while also learning rich embeddings that successfully improve downstream tasks like link prediction.

Peer-set analysis: For understanding the peer-set better, we isolate 3 merchants belonging to 3 separate industries and analyze their neighbours. Namely Skechers belonging to the Apparel Industry, Burger King belonging to the Food Industry, and Vitamin Shoppe belonging to the Stores Industry. From Figure reffig:peer(a), we can clearly observe the merchants belonging to three different industries are very well separated and close to one another(within industry) in the CaPE embedding space. Figure 3(b),(c), and (d) indicate a direct correlation between the distance in the real world (as determined by the latitude and longitude) and the L2 norm distance in the embeddings space, i.e., the merchants that are closer in physical world are also closer in embedding space and vice-versa.

Clustering analysis: Kmeans clustering is performed on the embeddings generated from GraphSAGE and CaPE and further visualised in Figure 4 using t-SNE in two dimension. It is observed that the embeddings are better clustered visually and quantitatively through the silhouette score.

5.2 Performance on Downstream Task

We also performed experimentation on link prediction task to compare CaPE embeddings with other state-of-the-art GNN algorithms. For the link prediction we perform a 80-20 train/test split and sample equal number of random negative edges (i.e. edges that do no exist in the original graph. Area Under the Receiver Operating Characteristic Curve (AUC) and Accuracy (ACC) are the primary

Method	Internal Data				Elo Merchant Data					
	Common Users	Category Preserved Industry	Distance (kms)	Spend Deviation	Common Users	Category Preserved Category	City	Important Feat Deviation Numerical 1	Numerical 2	Spend Deviation
GraphSAGE	1.01x	7.68%	**11.73**	44.78%	**2.30x**	49.69%	71.82%	419.77%	185.26	7.02%
GAT	0.93x	5.96%	11.45	55.86%	1.09x	52.39%	44.72%	496.88%	248.93%	6.64%
BiGI	1.03x	7.68%	11.82	44.95%	2.28x	17.78%	70.34%	527.91%	254.07%	7.84%
CaPE	**1.09x**	**7.84%**	**11.78**	**35.05%**	2.17x	**66.12%**	**72.72%**	**268.34%**	**125.24%**	**5.74%**

Table 1: Comparison of performance of CaPE with the baseline algorithms for Merchant Similarity Analysis. CaPE produces superior results than the baselines for most of the matrices.

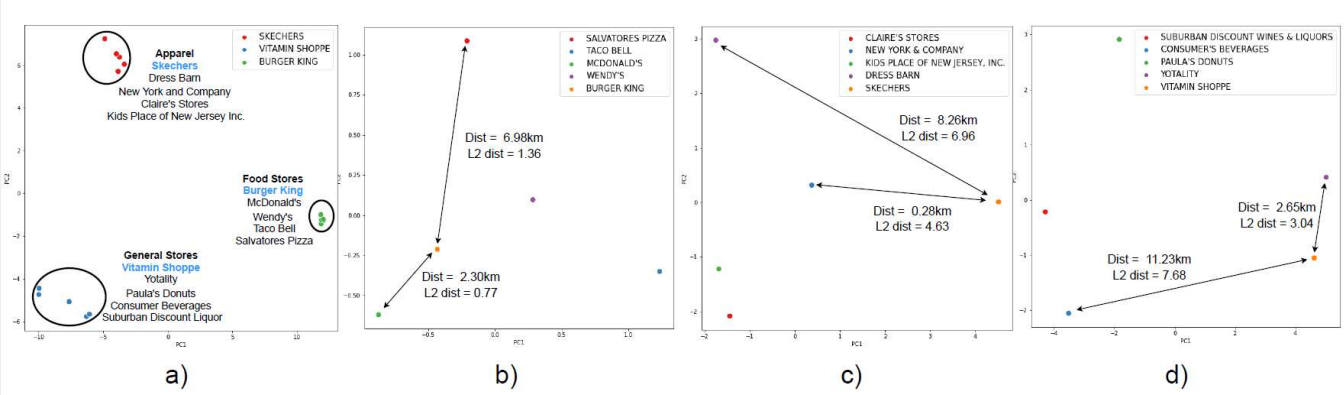

Figure 3: CaPE Merchant Peer Set Analysis

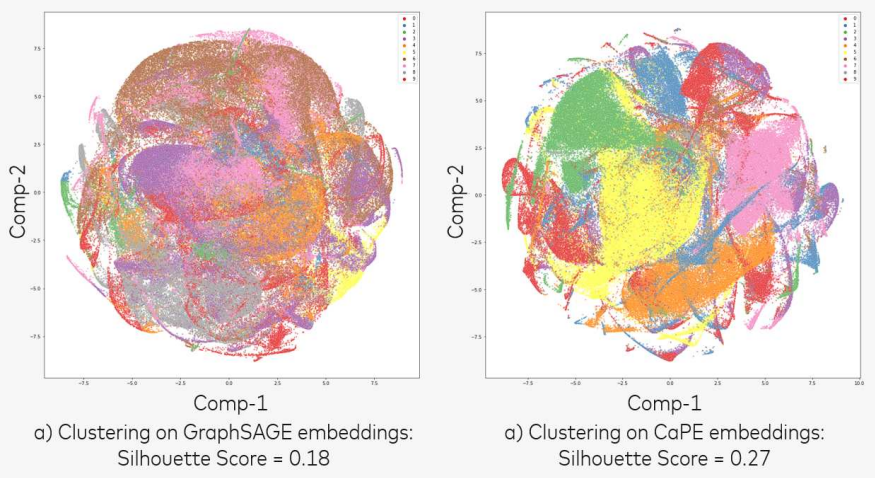

Figure 4: Clustering on embeddings using GraphSAGE vs CaPE

metrics for comparing the superiority of our algorithm. In Table 2, we can observe a 5-6% boost in AUC and a 2-5% boost in terms of ACC on our Internal Data as well as on Elo dataset. It shows that the embeddings generated from CaPE can be used for downstream applications as well.

6 CONCLUSION AND FUTURE WORK

In this paper, we present CaPE, a framework to generate category-preserving embeddings which also preserved important numerical feature information. Our framework solves a problem of generating embeddings for real-world datasets in financial domain having high cardinality categorical features. We introduce a mechanism to solve similarity-search problem in the payment industry using homogenous graph neural networks. We also introduce notion for measuring merchant similarity by proposing four similarity metrics that are relevant to the payement industry. We performed our experimentation on real-world dataset, both internal as well external to showcase the efficacy of our framework on similarity-search tasks as well as link prediction task. As future work, we plan

Method	Internal Data		Elo Data	
	AUC	ACC	AUC	ACC
GraphSAGE	0.94	0.88	0.76	70.90
GAT	0.88	0.83	0.81	**76.70**
BiGI	0.90	**0.82**	0.75	68.30
CaPE	**0.96**	**0.90**	0.82	75.80

Table 2: Comparison of performance of CaPE with the baseline algorithms for Link Prediction Task.

	Nodes	Edges	Feat Dim
Internal Data	212k	41M	97
Elo Data	100k	1.5M	14

Table 3: Dataset Statistics

Hyper-Parameters	Values
Neighbourhood Sample Size	[25, 10]
Embedding Dimension	128
Batch Size	512
Optimizer	Adam
Activation func	ELU
Learning Rate	0.0001

Table 4: Experiment Settings

to address the challenge of scalability by further optimizing our algorithm to perform similarity search at a scale of billions of edges and millions of nodes.

REFERENCES

[1] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[2] Belur V Dasarathy. 1991. Nearest neighbor (NN) norms: NN pattern classification techniques. *IEEE Computer Society Tutorial* (1991).

[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016), 3844–3852.

[4] Kaize Ding, Yichuan Li, Jundong Li, Chenghao Liu, and Huan Liu. 2019. Feature interaction-aware graph neural networks. *arXiv preprint arXiv:1908.07110* (2019).

[5] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 135–144.

[6] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292* (2015).

[7] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. 2018. Bine: Bipartite network embedding. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 715–724.

[8] Mihajlo Grbovic and Haibin Cheng. 2018. Real-time personalization using embeddings for search ranking at airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 311–320.

[9] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[10] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[11] Chaoyang He, Tian Xie, Yu Rong, Wenbing Huang, Junzhou Huang, Xiang Ren, and Cyrus Shahabi. 2019. Cascade-BGNN: Toward Efficient Self-supervised Representation Learning on Large-scale Bipartite Graphs. *arXiv preprint arXiv:1906.11994* (2019).

[12] Vassilis N Ioannidis, Da Zheng, and George Karypis. 2020. PanRep: Universal node embeddings for heterogeneous graphs. (2020).

[13] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data* (2019).

[14] Anish Khazane, Jonathan Rider, Max Serpe, Antonia Gogoglou, Keegan Hines, C Bayan Bruss, and Richard Serpe. 2019. Deeptrax: Embedding graphs of financial transactions. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 126–133.

[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[16] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[17] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, S Hamid Rezatofighi, and Silvio Savarese. 2019. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. *arXiv preprint arXiv:1907.03395* (2019).

[18] Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. 2019. Pytorch-biggraph: A large-scale graph embedding system. *arXiv preprint arXiv:1903.12287* (2019).

[19] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. 2017. Feature selection: A data perspective. *ACM computing surveys (CSUR)* 50, 6 (2017), 1–45.

[20] Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. 2014. Autoencoder for words. *Neurocomputing* 139 (2014), 84–96.

[21] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[22] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *International conference on machine learning*. PMLR, 2014–2023.

[23] Karl Pearson. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2, 11 (1901), 559–572.

[24] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[25] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM international conference on web search and data mining*. 555–563.

[26] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[28] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 839–848.

[29] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.