Predicting Economic Growth by Region Embedding: A Multigraph Convolutional Network Approach

Bo Hui
bohui@auburn.edu
Auburn University

Da Yan
yanda@uab.edu
University of Alabama at Birmingham

Wei-Shinn Ku
weishinn@auburn.edu
Auburn University

Wenlu Wang
wenluwang@auburn.edu
Auburn University

ABSTRACT

With the rapid progress of global urbanization and function division among different geographical regions, it is of urgent need to develop methods that can find regions of desired future function distributions in applications. For example, a company tends to open a new branch in a region where the growth trend of industrial sectors fits its strategic goals, or is similar to that of an existing company location; while a job hunter tends to search regions where his/her expertise aligns with the industrial growth trend providing sufficient job opportunities to sustain future employment and job-hopping.

Our solution is to learn a distribution (aka. embedding) of the growth of various industrial sectors for each region, so that the embeddings of different regions can be searched, or compared for similarity querying. We consider the fine granularity of ZIP code areas as they are usually representative of the regional functions. By effectively utilizing open data on the Internet such as government data (e.g., from US Census Bureau) and third-party data for supervised learning, we propose to first construct a multigraph that captures the various relationships between regions such as direct flight connections and shared school districts, and then learn region embeddings using a novel graph convolutional network architecture. Our multigraph convnet (MGCN) differentiates various feature types such as demographic, social, economic and housing features, and learns different weights on different features and spatial relationships for effective data-driven feature aggregation.

While deep learning is known to require large amounts of data to train, our weighted MGCN (WMGCN) is designed to minimize the number of parameters so that it does not underfit on the limited amount of open data. Extensive experiments are conducted to compare our WMGCN model with several competitive baselines to demonstrate the superiority of our WMGCN design.

CCS CONCEPTS

• **Information systems → Data mining**.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411882

KEYWORDS

Graph convolutional network; multigraph; embedding; geographical region; economic growth

ACM Reference Format:
Bo Hui, Da Yan, Wei-Shinn Ku, and Wenlu Wang. 2020. Predicting Economic Growth by Region Embedding: A Multigraph Convolutional Network Approach. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411882

1 INTRODUCTION

The Problem. We study the problem of embedding each ZIP code area into a vector representation that reflects its economic growth trend, which can be used for searching for a desired ZIP code area for planning future investments. For example,

- When a company wants to expand its businesses into new regions, it can search for a ZIP code area to set up its new site where the distribution of the growth of various industries is similar to that of an existing site so that the new site can benefit from prior success experience.
- When one changes his/her job and moves to a new work location, he/she can search for the ZIP code areas nearby to find a house with a similar growth trend of industrial sectors to purchase, e.g., school rating remains high, healthcare service remains strong, and/or finance industry is booming if the person and/or his/her spouse are in the financial industry.

US Census Bureau, which is the federal government's largest statistical agency, provides current facts and figures about America's people, places, and economy. These statistics are often released at different geographical levels such as states, counties, ZIP Code Tabulation Areas (ZCTAs), etc., on a yearly basis, and they are frequently used by policy makers and businesses for planning and decision making. In this paper, we also utilize these statistics to learn region embeddings.

We adopt the region granularity of ZIP code areas since different regions of a city can have very different functions and quality of life, which are important when people make investment decisions. For example, Whole Foods Market usually targets rich communities while Walmart is more affordable and may target dense residential areas. People with children may also want to buy a house in a good school district.

Table 1: NAICS Industry Sectors

Code	Industry Title
11	Agriculture, Forestry, Fishing and Hunting
21	Mining
22	Utilities
23	Construction
31-33	Manufacturing
42	Wholesale Trade
44-45	Retail Trade
48-49	Transportation and Warehousing
51	Information
52	Finance and Insurance
53	Real Estate Rental and Leasing
54	Professional, Scientific, and Technical Services
55	Management of Companies and Enterprises
56	Administrative and Support and Waste Management and Remediation Services
61	Educational Services
62	Health Care and Social Assistance
71	Arts, Entertainment, and Recreation
72	Accommodation and Food Services
81	Other Services (except Public Administration)
92	Public Administration

Why Economic Growth? Site selection is often not a short-term investment, and so it is important to ensure future growth. The current mass of an industry may not be sufficient.

For example, Rochester, NY is a city known as a home for educated engineers and Kodak was one of the biggest employers of the city's college graduates, thanks to its mass market for photography; however, with the popularity of digital cameras and smart phones, the industry is actually shrinking and Kodak had to file for bankruptcy protection in 2012. Regional impacts include forcing some universities such as Rochester Institute of Technology to shift emphasis from college education to more on research.

As another example, consider a tech startup in Alabama (e.g., Fledging[1]) which wants to find a location within the state to set up its office. Such a restriction is reasonable, for example, because the investors are local businessmen in Alabama. An ideal location would show growth of startups and available IT workforce (e.g., graduated from nearby universities), as in Silicon Valley. Even though the mass of startups and IT workforce in Alabama cities is much smaller than in Silicon Valley, the startup can still find a region in Alabama with the quickest growth of startups and IT workforce (i.e., most similar to Silicon Valley), such as Innovation Depot[2] in Birmingham, AL where the University of Alabama at Birmingham is located.

Targeted Region Representation. Unlike existing embedding methods where elements of the learned vector representations do not carry physical meanings per se, we require the dimensional vector of a zip code in our region embedding to represent the economic growth trend in the zip code area. This design allows people to search regions more intuitively, e.g., by searching nearest neighbors in the embedding space. Meanwhile, region similarities in terms of economic growth can be estimated from distance metrics of their embeddings. Finally, regions in the search results are directly interpretable using their embeddings.

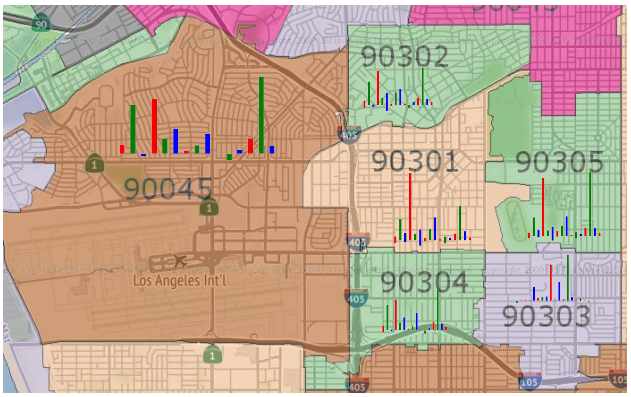

Figure 1: Sector-by-Sector Growth Vectors at ZIP Code Level

One challenge to train a region representation is the need of ground-truth growth values of different industrial indicators to provide necessary supervision. This is different from the skip-gram model widely used in embedding words [23] and graph nodes [13, 25] which uses the current word (or node) to predict its context, i.e., surrounding words in a sentence (or surrounding nodes in a random walk). Unlike our problem setting, the skip-gram model derives supervision directly from large corpus or many random walks in an unsupervised manner, and thus the training data are abundant.

Fortunately, the North American Industry Classification System (NAICS) divides the businesses into 20 sectors [3] as shown in Table 1: in each year and for each sector, the number of establishments at the ZIP code level within the US can be collected using US Census Bureau's ZIP Codes Business Patterns (ZBP) APIs [4]. This allows us to obtain the number of new establishments created in a ZIP code area between Year 2011 and Year 2016, for example, by subtracting the establishment counter vector of the latter by that of the former. In this paper, we use such a difference vector of each ZIP code area as its target embedding to learn. Intuitively, similar zip code areas in terms of economic growth will have similar industrial distributions of establishments' growth in the future. Altogether we collected data of 33119 ZIP codes within the US. Figure 1 illustrates our sector-by-sector ZIP code area embeddings.

Input Region Features. To predict the sector-by-sector economic growth vector of a region, we use the region's current features as an input source. American Community Survey (ACS) [1] from US Census Bureau provides such regional features in a yearly basis at different geographical levels. For each of the 33119 ZIP code areas, we collected 4 categories of features, including 82 demographic features (e.g., the total population and citizens), 150 social features (e.g., total households and fertility), 114 economic features (e.g.,

[1] https://fledging.net/
[2] https://innovationdepot.org/

income and benefits and population in the labor force), 142 housing features (e.g., average house value and total housing units).

Integrating Geographical Relations. Using region-specific features alone ignores the geographical relations between regions, which may miss important region similarity factors. For example, the fact that an airport located in a nearby ZIP code area can have a profound impact on the current ZIP code area's economic growth than on one that is farther away from the airport. Also, cities with frequent direct flights can share more economic growth opportunities than cities that are not directly connected by flights. As another example, parents may consider all ZIP code areas in the same good school district when purchasing a house; in other words, spatial units slightly larger than ZIP code areas could be of interest. Finally, a county usually only has a couple of Costco warehouse stores visited by residents from many ZIP code areas in the county.

We, therefore, propose to capture these diverse relations between ZIP code areas with a multigraph where each edge refers to one particular relationship between two areas. The features from neighboring areas of a ZIP code area can be used to enhance its features for better prediction, which is critical for a realistic embedding as ZIP code areas are usually too fine-grained: for example, residents may drive to a Costco store or airport in nearby area in 10 minutes, which does not compromise their preference for the region's economic sectors.

Approach Overview. Since spatial smoothing is important for ZIP code level region embedding, we achieve this by applying graph convolutional network (GCN) [17] over the previously described multigraph where nodes are ZIP code areas containing 4 different kinds of features, and edges capture different types of spatial relationships between two ZIP code areas. In GCN, each graph convolution layer propagates the features of one-hop neighbors. Our approach effectively executes a spatial regularization based on Tobler's first law of geography stating that "near things are more related than distant things", to battle the spatial feature discontinuity caused by the fine granularity of ZIP code area units.

Even though we are able to construct such a multigraph of ZIP code areas from open data, and use NAICS industry sector growth vector for supervised representation learning, the amount of data is still very limited compared with skip-gram models. Specifically, we only have 33119 ZIP codes within US, and we should only use the feature data and growth supervision from recent years to avoid out-of-date growth trend. This poses another challenge to building our embedding model, since the loss objective function of a deep learning model is typically non-convex and requires massive data to train the large quantity of weight parameters. In contrast, we only have a moderate-sized multigraph and thus a simple model with fewer parameters is preferred to avoid model underfitting, according to the principle of Occam's razor.

We thus adopt a simple GCN design to extract intermediate features from each category of region input features, which are then integrated to predict the final sector-by-sector growth region embeddings. After testing various ways of feature integration, we find that a simple weighted sum works the best (e.g., as compared to adding another dense layer) thanks to its simplicity to train which aligns with Occam's razor.

Contributions. Our main contributions are as follows:

- We identify open data sources (including government data and third-party data) to effectively construct our region-based multigraph for learning sector-by-sector region embeddings from the perspective of economic growth.
- We follow the principle of Occam's razor to design simple GCN based models that are easy to train on our limited data source without model underfitting, including techniques such as separating feature extraction of different categories of input region features, feature integration by weighted sum, and selection of top-k weighted edges for neighbor feature integration (see Section 4 for details).
- We evaluate our method on real data by comparing it extensively with existing alternative solutions to demonstrate its effectiveness.

Paper Organization. The rest of this paper is organized as follows. Section 2 reviews the related work regarding graph neural networks and region embedding. Section 3 then defines our notations and presents how our multigraph of ZIP code areas is constructed from open data. Section 4 introduces our model design and explains why it is more effective than other alternatives. Finally, Section 5 reports our experimental results and Section 6 concludes this paper.

2 RELATED WORK

2.1 Node Embedding & Graph Neural Networks

Topology-Based Node Embedding. Since deep learning layers take fixed-size tensors (or, a mini-batch of fixed-length vectors) as the input, a number of works propose to embed nodes in a graph into vectors so that graph data become admissible to deep learning tasks such as node classification and graph classification. For example, DeepWalk [25] traverses a graph to generate random walks to capture local structures by neighborhood relationships, and then uses the skip-gram model to learn the embedding of each node that is most likely to generate its nearby nodes (or their embeddings) in a walk. This is similar to word2vec [23] except that now sentences become walks and words become nodes. Node2vec [13] further introduces Breadth-Fast-Sampling (BFS) and Depth-First-Sampling (DFS) to control the random behavior. BFS reaches immediate neighbors while DFS prefers nodes away from the source.

A problem with using DeepWalk and node2vec in our setting is that, these methods only consider graph topology and cannot incorporate input node features. However, in our multigraph, edges only reflect the connection strength between different ZIP code areas, and it is the input node features that propagate along these edges to mutually reinforce each other w.r.t. economic factors.

Graph Neural Networks (GNNs). The notion of GNN was initially outlined in [12] and further elaborated by [27] as a form of recurrent neural network (RNN). They propagate neighbor information in an iterative manner until reaching a stable fixed point. This process is expensive and is recently improved by several studies such as [21]. Encouraged by the success of convolutional neural networks (CNNs) in computer vision which extracts higher-level features from images using a sequence of interleaving convolution layers and pooling layers, recent models focus on adapting these layers to work directly on graph input (rather than tensor input).

Graph Convolutional Networks (GCNs). Graph convolution layers can be divided into two categories, spectral graph convolution and localized graph convolution [32].

Early works mainly focus on spectral graph convolutions, as pioneered by [6]. In a follow-up work, ChebyNet [8] defines graph convolutions using Chebyshev polynomials to remove the expensive Laplacian eigendecomposition. Then, GCN [17] as the state of the art, further simplify graph convolution by a localized first-order approximation, leading to a simple form of

$$H^{(\ell+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(\ell)}W^{(\ell)}),$$

where $\tilde{A} = A+I$ is the adjacency matrix with self-connections added, $\tilde{D}$ is a diagonal matrix with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $W^{(\ell)}$ is the weight matrix to train for Layer ℓ, σ is an activation function such as ReLU, and $H^{(\ell)}$ is the state matrix where each row keeps the features of a node at Layer ℓ; $H^{(0)} = X$, i.e., the input matrix where each row keeps the features of a node.

However, spectral methods require operating on the entire graph Laplacian during training which is expensive, though some follow-up works endeavor to mitigate the cost as in FastGCN [7] and SGC [31]. To avoid operating on graph Laplacian, GraphSAGE [14] proposes to learn a function $f(\cdot)$ that generates node embeddings by aggregating features from a node's local neighborhood. Specifically, each convolution layer performs two transformations on each node v:

$$\mathbf{h}^{(\ell)}_{\mathcal{N}(v)} \leftarrow f(\{\mathbf{h}^{(\ell-1)}_u \mid \forall u \in \mathcal{N}(v)\}),$$
$$\mathbf{h}^{(\ell)}_v \leftarrow \sigma(\mathbf{W}^{(\ell)} \cdot \text{CONCAT}(\mathbf{h}^{(\ell-1)}_v, \mathbf{h}^{(\ell)}_{\mathcal{N}(v)}) + \mathbf{b}^{(\ell)}),$$

where we denote $\mathcal{N}(v)$ as v's neighborhood. Specifically, we first aggregate features of v's neighbors u, i.e., $\mathbf{h}_u$ from the previous layer by function $f(\cdot)$; then the aggregated features are concatenated with the old features of v itself, which then pass through a dense layer to obtain v's new features at Layer ℓ. GraphSAGE proposes 3 different choices of aggregating function $f(\cdot)$: mean aggregator, LSTM aggregator and pooling aggregator. For example, the mean aggregator simply averages the features of v and its neighbors.

PinSage [32] further improves the performance of localized graph convolution by only incorporating top-k nodes that exert the most influence on node v into $\mathcal{N}(v)$ rather than using the full neighborhood, which is selected based on node visit frequencies of short random walks starting from v. PinSage also adopts a different aggregation function from GraphSAGE:

$$\mathbf{h}^{(\ell)}_{\mathcal{N}(v)} \leftarrow \text{AVG}(\{\text{ReLU}(\mathbf{Q}^{(\ell)} \cdot \mathbf{h}^{(\ell-1)}_u + \mathbf{q}^{(\ell)}) \mid \forall u \in \mathcal{N}(v)\}),$$
$$\mathbf{h}^{(\ell)}_v \leftarrow \text{ReLU}(\mathbf{W}^{(\ell)} \cdot \text{CONCAT}(\mathbf{h}^{(\ell-1)}_v, \mathbf{h}^{(\ell)}_{\mathcal{N}(v)}) + \mathbf{b}^{(\ell)}).$$

Here, AVG(.) is weighted average where the weight for neighbor u equals the L_1 normalized visit counts of u by random walks from v. Note that different from GraphSAGE that directly averages neighbor features, PinSage transforms neighbor features $\mathbf{h}^{(\ell-1)}_u$ by a dense layer plus ReLU activation first before taking their weighted average, which introduces additional parameters $<\mathbf{Q}^{(\ell)}, \mathbf{q}^{(\ell)}>$ besides $<\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}>$ to each graph convolution layer, allowing more model capacity.

Graph Pooling. Besides convolution, pooling has been studied in the context of graph data as input, which reduces graph size for upper layers (hence reduces their parameter size). This is especially true when going from node embeddings to a representation of the entire graph for the task of graph classification, where global pooling is essential as in [9, 21]. SortPool [35] sorts nodes according to their structural roles within a graph, truncates the sorted embeddings to a fixed size k and then feeds them to the next layer.

Hierarchical pooling aims to learn hierarchical representations to capture structural information. DiffPool [33] coarsens each input graph by learning a differentiable soft assignment of nodes at Layer ℓ to clusters which are essentially nodes of Layer $(\ell + 1)$. gPool [11] achieves performance comparable to DiffPool with less cost. It uses a trainable projection vector $\mathbf{p}$ to project all node features to 1D, and then pick top-k nodes with the largest projected values to form a smaller graph. A gUnpool operation is also proposed so that a U-Net style encoder-decoder network can be constructed for node classification and graph classification. SAGPool [18] uses a self-attention layer to select top-ranked nodes to form a smaller graph, and the layer has the same form as a spectral convolution layer of GCN [17] though the activation function adopted is tanh.

2.2 Region Embedding

The problem of city similarity comparison was studied by [26] which simply models a city as a vector of counts of different venues, which are obtained from Foursquare check-in data. Two cities are compared using the cosine similarity of their vectors. This work finds that directly aggregated amenity counts in a city give slightly different similarity than if amenity counts are first aggregated by subareas in a city and then averaged, which hints that a city-level representation is too coarse to be stable. In fact, different regions in a city can have different economic features and functions. For example, the GDP of Manhattan Borough in New York City is one order of magnitude larger than that of Queens Borough, and [34] identifies different functional zones in Beijing through a topic model over human mobility trajectories. We thus consider the finer granularity of ZIP code areas in this paper.

The work of [29] goes beyond modeling a region simply with a frequency vectors of PoI (Point of interest) categories, to incorporate features that represent the spatial distribution of PoIs in the region. Specifically, the features are computed based on the distances of each object to five predefined reference points in the region. Since such hand-crafted spatial features do not consider the relative locations between objects, [22] addresses this problem by leveraging a deep learning model that is similar to FaceNet [28] to learn a ranking metric for comparing rectangular regions that are modeled as images with objects in grid cells (regarded as pixels).

The work of [10] uses Foursquare check-in data to derive features of venues (e.g., number of check-ins, number of likes) so that each venue is associated with a 30-dimensional feature vector. The work considers neighborhoods as regions, and each neighborhood is represented by its set of venues (and their feature vectors). Different neighborhoods are compared using Earth mover's distance, where the distance between venues is computed by a learned Mahalanobis distance of their feature vectors. Another work, [16], learns a generative process for the geotags inside a neighborhood from Flicker photos, and uses the distribution of sampling local tags of a neighborhood as its feature vector.

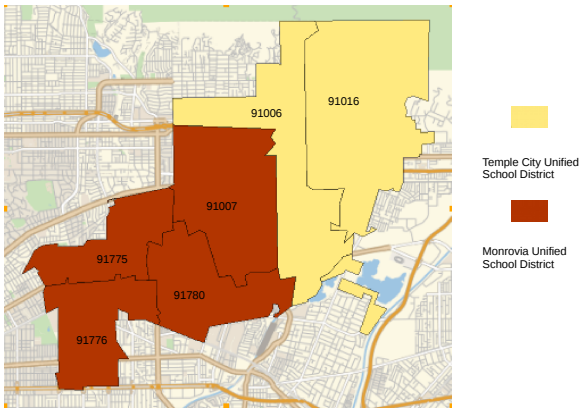

Figure 2: Illustration of Common School District

Our problem setting is different from those of the above works which utilize objects (e.g., PoIs, check-ins, geotags) inside a region to extract features, as our input features for each region are directly available from ACS which are related to the economy rather than PoI distribution or visiting preference. Also, we capture region relationships using a multigraph so that GCN can learn node (i.e., region) embeddings, while previous works directly consider the distance between region representations without resorting to a graph data model.

3 NOTATIONS AND THE MULTIGRAPH

A key difference of our work from those reviewed in Section 2.2 is that we not only consider the content information of each individual region, but also consider the structural relationships between regions. In this section, we define our notations and explain how we construct our multigraph over ZIP code areas; these definitions will provide sufficient context for understanding our methods to be described in Section 4.

We consider the geographic unit of ZIP code areas. Specifically, we denote the set of all ZIP code areas within US by $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{33119}\}$, where $\mathbf{x}_i$ is the input feature vector of the i-th ZIP code area. Recall that we collected 4 categories of features from American Community Survey (ACS) including 82 demographic features, 150 social features, 114 economic features, 142 housing features. We organize features in $\mathbf{x}_i$ accordingly into 4 segments:

$$\mathbf{x}_i = \text{CONCAT}(\mathbf{x}_{i,c_1}, \mathbf{x}_{i,c_2}, \mathbf{x}_{i,c_3}, \mathbf{x}_{i,c_4}),$$

where $\mathbf{x}_{i,c}$ refers to the segment of all features of the i-th ZIP code area that belong to Category c. We denote the set of all feature categories by $C = \{c_1, c_2, c_3, c_4\}$.

Besides the regions' own features, we also consider 3 kinds of spatial relations between two regions, which are modeled by 3 types of edges in our region multigraph. The first one is the school district relationship, where two regions are connected by such an edge if they belong to the same school district. Figure 2 shows an example of two common school districts, where ZIP code areas 91006 and 91016, for example, are in the same school district highlighted in yellow, and so they are connected by an edge.

We consider our multigraph as an undirected graph with weighted and typed edges, and the weight of a school-district edge between

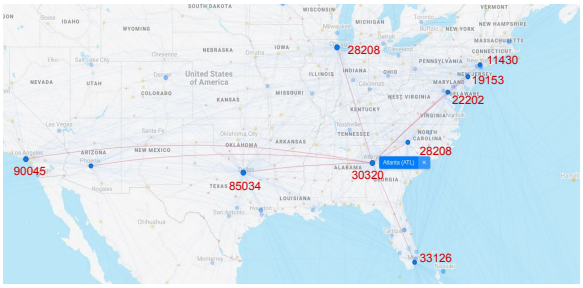

Figure 3: Illustration of the Relation of Direct Flights

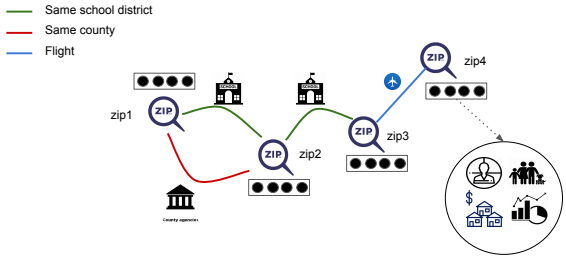

Figure 4: An illustration of the Multigraph

two ZIP code areas is computed as the reciprocal of the geographical distance between their internal points[3] [2]. This is based on Tobler's first law of geography. For example, in Figure 2, the edge between 91776 and 91007 has a lower weight than the edge between 91776 and 91780.

While the quality of a school district often affects the housing price and is a strong indicator of a region's economic characteristic, the county that a region belongs to is another important factor. For example, in Alabama, Jefferson County is the most populated county where Birmingham, the state's biggest city, is located, clustered by businessmen, medical and healthcare practitioners and people in the banking industry. The second most populated county is Mobile which contains Alabama's only saltwater port, and is considered one of the Gulf Coast's cultural centers and the most popular place for tourism for people in Alabama during holidays. The third most populated county is Madison County where the city of Huntsville is located, known as the home of engineers thanks to the fact that the largest NASA center, Marshall Space Flight Center, is located in Huntsville.

Therefore, the second spatial relation we consider is whether two ZIP code areas are in the same county, where edge weight is also computed as the reciprocal of the geographical distance between the internal points of the two regions considered. Sometimes a ZIP code area on county boundaries may overlap with more than one county, in which case it is linked by edges to nearby ZIP code areas from different counties.

A third important spatial relation is the flight connections between regions. An international airport can benefit the economics of surrounding areas, such as the Birmingham metropolitan area of Alabama covering 10 counties surrounding Birmingham-Shuttlesworth International Airport. In fact, if there are direct flights

[3] According to the glossary of the US Census Bureau [2], an internal point is calculated for each geographic entity as its geographic center.

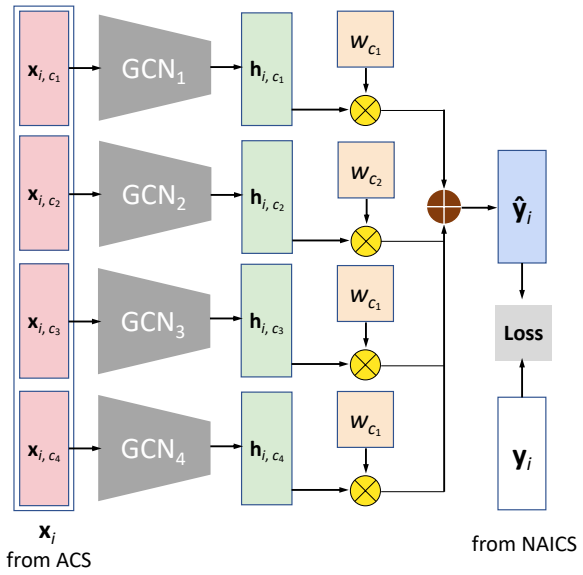

Figure 5: Model Overview

between two regions, these two regions will share great economic growth, and the number of daily flights reflects how closely they are connected. Therefore, we add an edge between two ZIP code areas if there are direct flights between them, and the edge weight equals the number of daily direct flights. See Figure 3 for an illustration of the edges.

Overall, we obtain a multigraph $G = (\mathcal{X}, E)$, where node set $\mathcal{X}$ are the ZIP code areas, and edge set $E = E_{school} \cup E_{county} \cup E_{flight}$ which consists of the 3 kinds of edges we described above. The result is a multigraph since two ZIP code areas could be connected by both a school-district edge and a same-county edge. See Figure 4 for an illustration of G.

We remark that we only model one-hop relations in G, since multi-hop relations can be captured by stacking GCN layers as we shall see in Section 4. Two-hop relations could improve performance as two regions with one-stop flight connects are also economically related, and nearby counties could have impacts over each other. But since our spatial relations are over much larger spatial units (school districts, counties, flight routes) than ZIP code areas, too many hops may cause oversmoothing in addition to the disadvantage of adding more cost and difficulty in model training.

4 THE MODEL

In this section, we present our model for embedding ZIP code areas, and explain why we make these design choices.

Figure 5 overviews our end-to-end model architecture, where we temporarily treat the GCN module as a black box for extracting intermediate region features from our multigraph. We will explain how we design our GCN shortly afterward.

Note that since our learning relies on open data that is still limited in size, our design principle is to use a simple model with fewer parameters to train, to make the model easy to learn and not underfitting following the principle of Occam's razor.

Feature Category Decoupling. For each ZIP code area $\mathbf{x}_i$ in a training mini-batch, we separate its features into 4 categories:

- $\mathbf{x}_{i,c_1}$ for Category c_1 with 82 demographic features;
- $\mathbf{x}_{i,c_2}$ for Category c_2 with 150 social features;
- $\mathbf{x}_{i,c_3}$ for Category c_3 with 114 economic features;
- $\mathbf{x}_{i,c_4}$ for Category c_4 with 142 housing features,

and for each category of features $\mathbf{x}_{i,c_j}$, we use GCN to extract a vector of intermediate features for this category, denoted by $\mathbf{h}_{i,c_j}$. Note that compared with working on $\mathbf{x}_i$ directly, we effectively reduce the number of parameters in our GCNs since, for example, when computing features $\mathbf{h}_{i,c_1}$, we know that we do not need to consider the path from $\mathbf{x}_{i,c_j}$ for $j = 2, 3, 4$ in the computation graph of our model.

Intuitively, here we are building a more accurate graphical model by utilizing the background knowledge that different feature categories are not dependent on each other, and similar benefits of this idea have been observed in other works such as [20]. As we shall see in Section 5, our decoupling of input feature categories when extracting features more effectively improves the embedding quality than if we extract intermediate features using a GCN by blindly treating $\mathbf{x}_i$ as a whole.

Weighted Feature Integration. Now that we have obtained intermediate features from the input features of all 4 categories, i.e., $\mathbf{h}_{i,c_1}, \mathbf{h}_{i,c_2}, \mathbf{h}_{i,c_3}$ and $\mathbf{h}_{i,c_4}$, a naïve approach is to concatenate them into a big feature vector $\mathbf{h}_i$, and then to use a dense layer (fully connected neural network) to transform $\mathbf{h}_i$ to our 20-dimensional vector $\hat{\mathbf{y}}_i$ to predict, which keeps the sector-by-sector predicted economic growth values.

However, this approach creates a large amount of weight parameters to train: let the length of each vector $\mathbf{h}_{i,c_j}$ be ℓ, then we have a weight matrix of size $4\ell \times 20 = 80\ell$ elements as the parameters of the dense layer.

In contrast, we make the assumption that a GCN by itself is powerful enough to directly learn the target 20-dimensional vector of predicted economic growth values from a multigraph with node features $\{\mathbf{x}_{i,c_j} \mid \forall \mathbf{x}_i \in \mathcal{X}\}$ from Category c_j. That is, we require that each vector $\mathbf{h}_{i,c_j}$ is already 20-dimensional with the sectors aligned to the vector positions. These 4 intermediate prediction vectors are then integrated into our final prediction $\hat{\mathbf{y}}_i$ simply using a weighted sum:

$$\hat{\mathbf{y}}_i \leftarrow w_{c_1} \cdot \mathbf{h}_{i,c_1} + w_{c_2} \cdot \mathbf{h}_{i,c_2} + w_{c_3} \cdot \mathbf{h}_{i,c_3} + w_{c_4} \cdot \mathbf{h}_{i,c_4},$$

which only has 4 trainable weight parameters $(w_{c_1}, w_{c_2}, w_{c_3}, w_{c_4})$, rather than the entire weight matrix of a dense layer. Besides fewer parameters to train, our weighted-sum scheme also has the physical implication that the growth contributions from different categories are additive.

In fact, such weighted-sum scheme has been successfully applied in other scenarios that require additivity. For example, [30] predicts the success rate of an itemset with different types of items by creating a feature vector for the itemset as a weighted sum of the feature vectors of individual items. The intuition is that whether a paper is accepted depends additively on the various items like authors, keywords, references cited, etc. We are different in that [30] treats their weights as hyperparameters to tune, while our weight parameters $(w_{c_1}, w_{c_2}, w_{c_3}, w_{c_4})$ are directly computed by end-to-end training.

Loss Function. The goal of our model is to predict the ground-truth sector-by-sector economic growth vector $\mathbf{y}_i$, and thus our

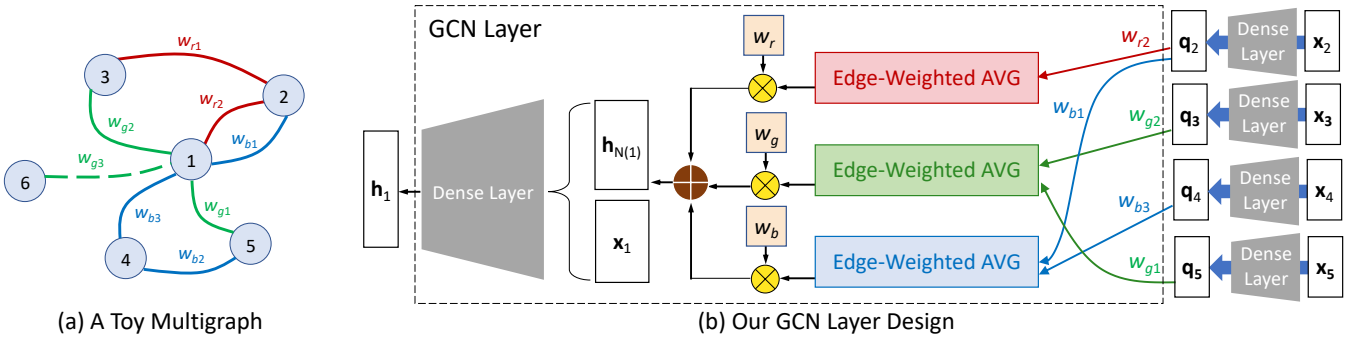

Figure 6: GCN Design

loss function for region $\mathbf{x}_i$ is defined as the difference between our prediction $\hat{\mathbf{y}}_i$ and the actual $\mathbf{y}_i$, measured in L_2-norm:

$$\mathcal{L}_i = \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_2^2,$$

and our overall loss function is simply summing the per-region losses for all regions in a mini-batch. Next, we give some analysis based on our weighted-sum feature integration design.

We can rewrite the loss as (where $<.,.>$ is the inner product):

$$\begin{aligned}\mathcal{L}_i &= (\hat{\mathbf{y}}_i - \mathbf{y}_i)^T(\hat{\mathbf{y}}_i - \mathbf{y}_i) \\ &= \|\hat{\mathbf{y}}_i\|_2^2 + \|\mathbf{y}_i\|_2^2 - 2<\hat{\mathbf{y}}_i, \mathbf{y}_i>,\end{aligned}$$

and since we have defined $\hat{\mathbf{y}}_i = \sum_j w_{c_j}\mathbf{h}_{i,c_j}$, the first term of $\mathcal{L}_i$ can be written as:

$$\begin{aligned}\|\hat{\mathbf{y}}_i\|_2^2 &= (\sum_j w_{c_j}\mathbf{h}_{i,c_j}^T)(\sum_k w_{c_k}\mathbf{h}_{i,c_k}) \\ &= \sum_j w_{c_j}^2\|\mathbf{h}_{i,c_j}\|_2^2 + 2\sum_{j,k:j<k} w_{c_j}w_{c_k}<\mathbf{h}_{i,c_j}, \mathbf{h}_{i,c_k}>,\end{aligned}$$

so to minimize the loss, we try to reduce $<\mathbf{h}_{i,c_j}, \mathbf{h}_{i,c_k}>$ which pushes our different GCN modules (over different ACS feature categories) to learn different aspects of economic growth, so as to be more specialized. Also, $\|\mathbf{h}_{i,c_j}\|_2^2$ serves as a regularization term to avoid value exploding in any $\mathbf{h}_{i,c_j}$.

In contrast, the third term of $\mathcal{L}_i$ can be written as:

$$\begin{aligned}-2<\hat{\mathbf{y}}_i, \mathbf{y}_i> &= -2<\sum_j w_{c_j}\mathbf{h}_{i,c_j},\ \mathbf{y}_i> \\ &= -2\sum_j w_{c_j}<\mathbf{h}_{i,c_j},\ \mathbf{y}_i>,\end{aligned}$$

so to minimize the loss, we try to increase $<\mathbf{h}_{i,c_j}, \mathbf{y}_i>$ which pushes the per-category 20-dimensional prediction $\mathbf{h}_{i,c_j}$ to be close to the ground truth $\mathbf{y}_i$.

To summarize our analysis, our loss function combined with our weighted-sum feature integration scheme allows our training to consider not only the contribution division and regularization of intermediate feature vectors, but also the target prediction vectors' similarity to the ground truth.

GCN over the Region Multigraph. In Figure 5, we have 4 GCNs, one to be trained on each category of ACS input region features. We remark that the 4 GCNs are operating on the same multigraph topology we introduced in Section 3, and the only difference is on region-nodes' input features.

Recall that we require each GCN to have sufficient model capacity to learn the targeted sector-by-sector embeddings. Therefore, we adopt PinSage's localized graph convolution scheme as we have reviewed in Section 2.1, which has two sets of learnable weight parameters $<\mathbf{Q}, \mathbf{q}>$ and $<\mathbf{W}, \mathbf{b}>$ at each layer: (1) a dense layer parameterized with $<\mathbf{Q}, \mathbf{q}>$ to transform neighboring features before aggregation, and (2) another dense layer parameterized with $<\mathbf{W}, \mathbf{b}>$ to transform the aggregated feature. We choose PinSage's GCN scheme [32] since it is more recent and well verified in industrial applications.

However, the unique characteristic of our problem setting is that our multigraph has three different types of edges, and their weight scales can be very different and incomparable: for school-district and common-county edges, the weights are the reciprocal of distance, while for flight edges, the weights are the number of daily flights. Moreover, the importance of different edge types could also be very different.

Therefore, we need to adapt PinSage's graph convolution layer to account for the edge heterogeneity of our multigraph. Figure 6(b) illustrates our design of one GCN layer where we specifically show how we extract the intermediate features of ZIP code area $\mathbf{x}_1$ (i.e., the output feature vector $\mathbf{h}_1$) from its neighbors in the toy multigraph in Figure 6(a).

We remark that each GCN of ours takes only one category of input region features (see Figure 5), i.e., $\mathbf{x}_{i,c_j}$ for Category c_j. However, for ease of presentation, we ignore notation c_j in Figure 6(b) and abuse our notation to directly use $\mathbf{x}_i$ (not to be confused with the entire input features including all our 4 categories).

In Figure 6(a), we use colors red, blue and green to denote our 3 types of edges: school-district, common-county, flight-connection, respectively. Considering all the neighbors of a region-node not only increases a model's training cost, but also introduces noisy features from less related regions which negatively impacts our smoothing quality, and therefore we only pick top-k most related neighboring regions for smoothing. This k-nearest-neighbor strategy is used by many GNN studies [19, 32] as well as other applications such as indoor localization [24].

Recall from Section 2.1 that PinSage uses random walks to select the most influential neighbors of a vertex for smoothing by GCN. However, PinSage's solution is not applicable in our setting since our graph edges are heterogeneous and the edge weights cannot properly reflect the transition probability beyond their relative order of importance.

We propose a voting strategy instead of those commonly used by ensemble models. Specifically, for each edge type, we consider the top-k neighbors of a node where the edge weights are the highest, and then integrate these nodes' feature vectors by a weighted-average where weights are simply the edge weights, and as in PinSage, nodes' feature vectors $\mathbf{x}_i$ are first transformed by a dense layer into $\mathbf{q}_i$ before the weighted aggregation (see Figure 6(b)).

Consider, for example, Figure 6(a) where we consider Node 1 and pick top-2 neighbors for smoothing in each edge category. While Nodes 3, 5 and 6 are all connected to Node 1 with green edges, we do not incorporate Node 6 in our smoothing in Figure 6(b) (see the green box) since the edge weight w_{g3} is less than w_{g1} and w_{g2} and we only consider top-2 neighbors. In other words, Node 1's integrated feature vector from its top-2 neighbors connected by green edges is given by $(w_{g2}\mathbf{q}_3 + w_{g1}\mathbf{q}_5)/(w_{g2} + w_{g1})$, where $\mathbf{q}_3$ and $\mathbf{q}_5$ are transformed representations of $\mathbf{x_3}$ and $\mathbf{x_5}$ by an initial dense layer. Note that w_{g1} and w_{g2} have the same scale and thus the weighted average makes good sense, similar to the mean aggregator of GraphSAGE [14].

Now that we obtain the edge-weight integrated neighbor feature vectors from all the 3 edge relations, denoted by $\mathbf{h}_r$, $\mathbf{h}_g$ and $\mathbf{h}_b$, we use a weighted sum to aggregate them into Node 1's neighbor features:

$$\mathbf{h}_{\mathcal{N}(1)} = w_r\mathbf{h}_r + w_g\mathbf{h}_g + w_b\mathbf{h}_b,$$

where weights (w_r, w_g, w_b) are learnable parameters to balance the contributions of features from different edge relations, to be tuned end-to-end. We remark that using weights effectively simplifies our model training, which is following the same idea as now we integrate features outputs from different GCNs as shown in Figure 5.

The second half of the graph convolution layer is the same as in PinSage and GraphSage: we concatenate the aggregated neighbor features $\mathbf{h}_{\mathcal{N}(1)}$ with Node 1's own features $\mathbf{x}_1$, and the resulting feature vector is then transformed by a dense layer to the output feature vector $\mathbf{h}_1$ of the current GCN layer (for Node 1).

To consider multi-hop neighbors, we can stack the previous GCN layers to extract higher-level features. Assume that we have extracted features $\mathbf{h}_1$ to $\mathbf{h}_6$ for Nodes 1 to 6 as in Figure 6(b); then we can input them in replace of $\mathbf{x}_1$ to $\mathbf{x}_6$ as the GCN layer's input at the 2nd layer, to extract new features $\mathbf{h}_1^{(2)}$ to $\mathbf{h}_6^{(2)}$; these features can then be returned for integration as shown in Figure 5. This two-layer GCN model considers 2-hop neighbors since, for example, $\mathbf{h}_2^{(2)}$'s computation uses $\mathbf{h}_1$, which further uses $\mathbf{x}_4$ of Node 4 which is 2 hops away from Node 2. We use multi-layer GCNs so that each GCN has sufficient capacity to learn good sector-by-sector growth predictions to be integrated by a simple weighted sum.

5 EXPERIMENTS

Data Collection. We adopt data from US Census Bureau which has been widely used for data mining [5]. Recall from Section 1 that we collected 33119 ZIP code areas released by ACS in 2011, which contains 4 categories of region features: 82 features on demographic, 150 on social, 114 on economic, 142 on housing collected in Year 2010; we also used the ZIP Codes Business Patterns API to collect the establishment growth counts from Year 2011 to Year 2016 for the 20 NAICS sectors, though growth counts between other recent years can also be used.

We also used the following open sources to collect edge relations between ZIP code areas:

- School districts: http://proximityone.com/zip-sd.htm;
- Counties: https://data.world/niccolley/us-zipcode-to-county-state;
- Flight routes: https://openflights.org/data.html#airline,

which were used to build our multigraphs. The number of flights we collected is 4897. We remark that the data is still limited which requires us to adopt a simple model design to avoid underfitting.

Evaluation Metrics. We evaluate the performance of our model and compare it with other baseline approaches using the following two metrics.

- **Mean Reciprocal Rank (MRR) [22]:** MRR is a rank-based metric defined as follows. Given a ZIP code area $\mathbf{x}_i$ to query, let $\mathbf{x}^*$ be the most similar ZIP code area to $\mathbf{x}_i$ in terms of economic growth (i.e., $\mathbf{x}^* = \arg\min_{\mathbf{x}_j \in \mathcal{X}-\mathbf{x}_i} \|\mathbf{y}_i - \mathbf{y}_j\|_2$), then we define $rank(\mathbf{x}_i)$ as the rank of $\mathbf{x}^*$ if we sort all the ZIP code areas $\mathbf{x}_j \in \mathcal{X} - \mathbf{x}_i$ by non-decreasing value of $\|\hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j\|$. If our prediction is accurate, $rank(\mathbf{x}_i)$ will be small.
  We define the Reciprocal Rank (RR) of query region $\mathbf{x}_i$ as

  $$RR(\mathbf{x}_i) = \frac{1}{\lceil rank(\mathbf{x}_i)/10 \rceil},$$

  so that if our prediction is accurate, $rank(\mathbf{x}_i)$ is small and thus $RR(\mathbf{x}_i)$ is large. The scaling factor 10 ensures that, for example, the difference between rank at 100 and rank at 200 is still noticeable, instead of being very close to 0.
  We report the average RR over a randomly selected test set Q of $n = 1000$ queries in our experiments, which is defined as the Mean Reciprocal Rank (MRR):

  $$MRR(Q) = \frac{1}{n}\sum_{\mathbf{x}_i \in Q} RR(\mathbf{x}_i).$$

- **Ratio of Triplets (RoT):** we sample region triplets [15] to evaluate our method. Given a region triplet $<\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3>$ ordered such that $\|\mathbf{y}_1 - \mathbf{y}_2\|_2 \leq \|\mathbf{y}_1 - \mathbf{y}_3\|_2$, we say that our embedding is successful if $\|\hat{\mathbf{y}}_1 - \hat{\mathbf{y}}_2\|_2 \leq \|\hat{\mathbf{y}}_1 - \hat{\mathbf{y}}_3\|_2$. In our experiment, we randomly generate 1000 triplets as our test set and compute the ratio of triplets (RoT) which are successfully embedded.

Both metrics have their emphasis: MRR evaluates the effectiveness of similar-region queries (e.g., if the top-10 result list contains a valid region that users are looking for), while RoT evaluates the overall quality of our embedding in preserving the similarity order among all the ZIP code areas.

Baselines. We adapt the existing work on geographical area recommendation that we reviewed in Section 2.2 to serve as our baseline region embedding competitors, including:

- **COS [29]:** this method directly compares the input feature vector $\mathbf{x}_i$ of two regions using cosine similarity;
- **Large Margin Nearest Neighbor (LMNN) [10]**: this method evaluates the similarity of two input region feature vectors using Mahalanobis distance, whose covariance matrix is learned from data using a method called Large Margin Nearest Neighbor (LMNN) described in [10];

Table 2: MRR and RoT Comparison: WMGCN v.s. Baselines

Algorithm	MRR	RoT
COS	0.1570	0.7107
LMNN	0.1705	0.6960
Triplet	0.1792	0.7922
PinSage	0.2475	0.8037
MGCN	0.2501	0.8107
WMGCN (1 GCN Layer)	0.2485	0.8277
WMGCN (2 GCN Layers)	**0.2755***	**0.8319***
WMGCN (3 GCN Layers)	0.2661	0.8123

- **Triplet [22]:** this method applies a triplet network to learn a ranking metric for comparing regions. The outputs are region embeddings that respect the similarity ranking;
- **PinSage [32]:** this is the only GCN baseline that considers both graph topology and node features. PinSage uses random walks to select influential neighbors for smoothing, so edge weights in our multigraph are ignored. We implemented PinSage with 2 graph convolution layers.

To justify our design, we also compare with the following variant of our Weighted Multigraph GCN (**WMGCN**) algorithm.

- **Multigraph GCN (MGCN):** this treats all input region features as an entire feature vector to be input to a single GCN for feature extraction. This is in contrast to our WMGCN that applies four GCNs over four different categories of input region features followed by a weighted-sum feature aggregation (see Figure 5);

Recall that WMGCN picks the top-k neighbors (according to edge weights) in each edge category for feature aggregation. To explore the effectiveness of smoothing from different neighborhood size, we also conduct experiments with different values of k.

To test the limit of our GCN layer, we also test the effect of stacking different numbers of GCN layers for feature extraction.

Model Configuration. For all the above algorithms, we use the following default design and hyperparameters unless otherwise stated. For our GCN (see Figure 6(b)), the default value of k is 2, i.e., we only aggregate features from the 2 nearest neighbors in each edge type. The feature vector size of any hidden layer is set to be 40. The models are trained using an SGD optimizer with a learning rate of 0.01 and a weight decay of 0.5. Other optimizer settings can be used but we find the difference in performance is insignificant.

Experimental Environment. All the algorithms were implemented in PyTorch 0.4.0. The experiments were conducted on a machine equipped with an NVIDIA Tesla P100 GPU, 26 GB RAM and a 2.20GHz CPU. Each of our experiments is repeated 3 times and the average results are reported to deal with the randomness of parameter initialization in training, though we find our results to be quite stable across different runs.

Algorithm Comparison. Table 2 shows the performance of our WMGCN (with different number of GCN layers) and the baseline algorithms for comparison.

We can obtain the following observations from Table 2:

- Models that leverage both region features and relations between regions, i.e., PinSage MGCN and WMGCN, outperform the other methods (COS, LMNN and Triplet) by a large margin, since the latter algorithms only consider region features. This demonstrates that smoothing effect by neighboring regions, as realized by GCN, is vital to the quality of region embedding.
- Our WMGCN model improves in both MRR and RoT as the number of GCN layers increases from 1 to 2, but going beyond 2 layers does not improve the performance further but actually backfires a bit due to the need to train more parameters (recall the principle of Occam's razor).
- Among different models that use 2 GCN Layers, our WMGCN model beats MGCN by a large margin thanks to its differentiation of different types of input region features.
- MGCN, in turn, beats PinSage by a large margin, thanks to the top-k neighbor smoothing technique and feature aggregation scheme from different spatial relation types; recall that PinSage simply finds top-k neighbors using random walks without considering edge types and edge weights in our multigraph. Our scheme exactly follows and thus respects Tobler's first law of geography that "near things are more related than distant things".

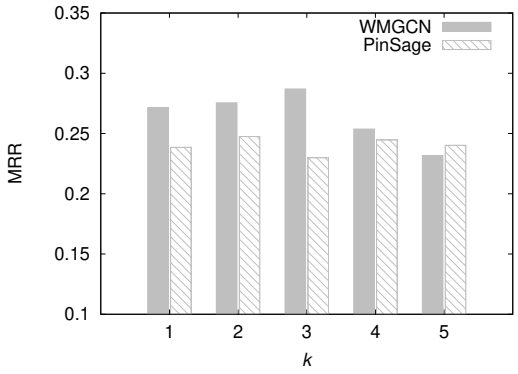

Figure 7: MRR with Different Values of k

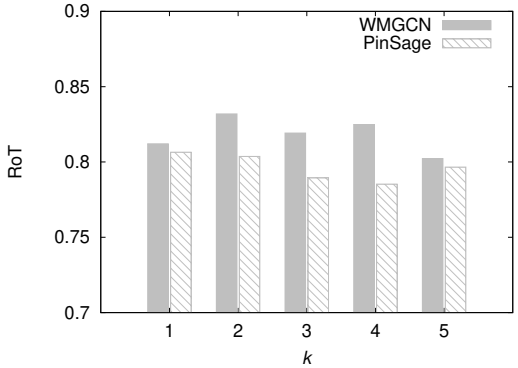

Figure 8: RoT with Different Values of k

Effect of Neighbor Smoothing. Recall that for each spatial relation (i.e., edge type), we smooth a region's features by integrating them with those of its top-k neighbors in our GCNs. Closer neighbors are more similar but more neighbors can better cancel the spatial discontinuity caused by the low granularity of ZIP code areas, and therefore, there is a tradeoff here and we want to conduct experiments to find the value of k that works the best in terms of our two metrics MRR and RoT.

We therefore vary the value of k from 1 to 5, and Figure 7 (resp. Figure 8) reports their performance w.r.t. MRR (resp. RoT). We can see that WMGCN with $k = 3$ achieves the best MRR, and that RoT is the highest when $k = 2$. Overall, 2 and 3 are good choices for k.

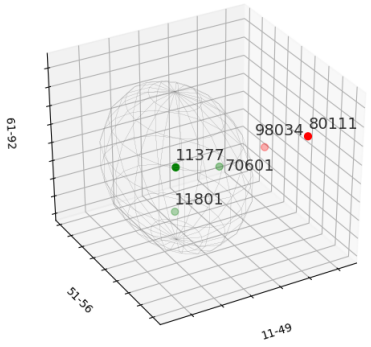

Figure 9: Nearest Neighbors of ZIP Code 11377

Visualization. We cannot directly visualize our embeddings in 20D space. Therefore, we group the 20 sectors in Table 1 into 3 groups by the sector code ranges 11–49, 51–56 and 61–92. For each 20D embedding vector, we compute the summation of the element values in each group, which gives us 3D coordinates of the form $(sum_{11-49}, sum_{51-56}, sum_{61-92})$ which can be visualized in 3D space to get a feeling of the distance between regions. To avoid scale difference of the 3 dimensions, we use min-max normalization to normalize $(sum_{11-49}, sum_{51-56}, sum_{61-92})$, and Figure 9 shows the 3D coordinates of 5 ZIP code areas.

Given a query ZIP code 11377, ZIP code areas 11802 and 70601 are more similar to 11377 than 98034 and 80111. The 3D visualization is, however, more intuitive, and sectors can be grouped in different ways depending on the needs.

6 CONCLUSION

We described a novel GCN-based model for learning economic growth sector-by-sector predictions from ACS features of ZIP code areas expanded with spatial relations. The design principle is to reduce the number of parameters to train, and to decouple the feature extraction process of different types of region and edge types. Experiments confirm that our learned region embeddings have a higher quality than existing baselines.

ACKNOWLEDGMENTS

This research has been funded in part by the U.S. National Science Foundation grants IIS-1618669 (III) and ACI-1642133 (CICI).

REFERENCES

[1] Visited in 2020. American Community Survey. https://www.census.gov/programs-surveys/acs.
[2] Visited in 2020. Internal Point. https://www.census.gov/programs-surveys/geography/about/glossary.html.
[3] Visited in 2020. NAICS. https://www.naics.com/business-lists/counts-by-naics-code/#countsByNAICS.
[4] Visited in 2020. ZIP Codes Business Patterns (ZBP) APIs. https://www.census.gov/data/developers/data-sets/cbp-nonemp-zbp/zbp-api.html.
[5] John M. Abowd. 2018. The U.S. Census Bureau Adopts Differential Privacy. In *SIGKDD*, Yike Guo and Faisal Farooq (Eds.). ACM, 2867.
[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.
[7] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR*. OpenReview.net.
[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*. 3837–3845.
[9] 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. In *NIPS*. 2224–2232.
[10] Géraud Le Falher, Aristides Gionis, and Michael Mathioudakis. 2015. Where Is the Soho of Rome? Measures and Algorithms for Finding Similar Neighborhoods in Cities. In *ICWSM*. AAAI Press, 228–237.
[11] Hongyang Gao and Shuiwang Ji. 2019. Graph U-Nets. In *ICML (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 2083–2092.
[12] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks*, Vol. 2. IEEE, 729–734.
[13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *SIGKDD*. ACM, 855–864.
[14] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*. 1024–1034.
[15] Elad Hoffer and Nir Ailon. 2015. Deep metric learning using Triplet network. In *ICLR, Workshop Track Proceedings*.
[16] Mohamed Kafsi, Henriette Cramer, Bart Thomee, and David A. Shamma. 2015. Describing and Understanding Neighborhood Characteristics through Online Social Media. In *WWW*. ACM, 549–559.
[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[18] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-Attention Graph Pooling. In *ICML (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 3734–3743.
[19] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*. AAAI Press, 3538–3545.
[20] Xia Li, Wei Li, Yuqun Zhang, and Lingming Zhang. 2019. DeepFL: integrating multiple fault diagnosis dimensions for deep fault localization. In *ISSTA*. ACM, 169–180.
[21] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In *ICLR*.
[22] Yiding Liu, Kaiqi Zhao, and Gao Cong. 2018. Efficient Similar Region Search with Deep Metric Learning. In *SIGKDD*. ACM, 1850–1859.
[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.
[24] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. 2004. LANDMARC: Indoor Location Sensing Using Active RFID. *Wireless Networks* 10, 6 (2004), 701–710.
[25] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *SIGKDD*. ACM, 701–710.
[26] Daniel Preotiuc-Pietro, Justin Cranshaw, and Tae Yano. 2013. Exploring venue-based city-to-city similarity measures. In *UrbComp@KDD*. ACM, 16:1–16:4.
[27] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Trans. Neural Networks* 20, 1 (2009), 61–80.
[28] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*. IEEE Computer Society, 815–823.
[29] Chang Sheng, Yu Zheng, Wynne Hsu, Mong-Li Lee, and Xing Xie. 2010. Answering Top-k Similar Region Queries. In *DASFAA (Lecture Notes in Computer Science, Vol. 5981)*. Springer, 186–201.
[30] Daheng Wang, Meng Jiang, Qingkai Zeng, Zachary Eberhart, and Nitesh V. Chawla. 2018. Multi-Type Itemset Embedding for Learning Behavior Success. In *SIGKDD*. ACM, 2397–2406.
[31] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 6861–6871.
[32] 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 *SIGKDD*. ACM, 974–983.
[33] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In *NeurIPS*. 4805–4815.
[34] Nicholas Jing Yuan, Yu Zheng, Xing Xie, Yingzi Wang, Kai Zheng, and Hui Xiong. 2015. Discovering Urban Functional Zones Using Latent Activity Trajectories. *IEEE Trans. Knowl. Data Eng.* 27, 3 (2015), 712–725.
[35] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An End-to-End Deep Learning Architecture for Graph Classification. In *AAAI*. AAAI Press, 4438–4445.