Bilateral Trade Flow Prediction Through GraphSAGE and MLP

Wilson Strasilla
Department of Computer Science
San Jose State University
San Jose, California
wilson.strasilla@sjsu.edu

***Abstract*—With international trade continuing to grow to higher levels, predicting trade amounts between nations becomes more important for planning. The gravity model has been used by economists to achieve this goal, but does not take into consideration many important factors. This paper proposes two solutions to improve upon past works to forecast international trade quantities, and potentially other datasets which follow a gravity-like pattern such as migration and traffic. First, a simple Multi-Layer Perceptron (MLP) is proposed which achieves its best results on feature rich data. Second, a model which uses GraphSAGE alongside MLP for prediction, and currently performs best on a smaller dataset of simple features, but can better apply to dynamic data than previous experiments using other Graph Neural Networks (GNNs) such as Graph Convolutional Networks (GCNs).**

***Index Terms*—Graph Neural Networks, International Trade, Gravity Model, GraphSAGE, Multi-Layer Perceptron, Link Analysis**

Code Access: https://github.com/wistrasy/CS274Project.git

I. Introduction

In recent years, international trade has continued to grow at very high rates. As a result, it becomes more important for governments and industries to be able to predict how much they will be importing and exporting to set policies and plans. In economics a common model used to predict how much two countries are likely to be trading is the gravity model of international trade which gets its name from its considerable parallels to Isaac Newton's law of gravitation [1]. This model predicts how much countries trade considering the size of the countries and their distance from each other as seen in (1). However, there are surely many other factors which determine how much trade goes between countries, such as resource availability and cultural similarities.

$$TradeFlow_{u,v} \approx \gamma \frac{GDP_u GDP_v}{distance(u,v)} \tag{1}$$

In addition to international trade, there are many other real world trends that follow this gravity like model such as migration and traffic patterns. Some of these applications require larger graphs to represent. For example, a city to city model will have many more data points than a country to country model. One of the hopes in studying this gravity model phenomenon is that a model that does well in one application can generalize to other similar problems such as those listed above.

This paper makes a few main contributions to this topic:

- A simple model is proposed, using a Multi-Layer Perceptron (MLP) to predict trade flow. This model performs better than previously used Graph Neural Network (GNN) reliant models for larger and more complex datasets.
- A model using GraphSAGE to embed nodes before making predictions with MLP is proposed. It can be seen that with this dataset the GraphSAGE performs better than the original Graph Convolutional Network (GCN), and should apply better to dynamic datasets.

II. Related Study

Gravity models have been studied in many different areas to better understand the links between two objects through measures of their sizes and distance. This includes research in the areas of migration [5], traffic flow [6], and of course international trade. Several Improvements have been made to trade forecasting through gravity informed machine learning techniques such as neural networks. Wohl et al. used shallow neural networks to handle this task in 2018 [3]. In 2020, Grubelich et al. found that decision trees outperformed Gaussian processes, Long Short Term Neural networks, and Convolutional Neural Networks in their experiments [4]. Recently, studies such as those of Panford-Quainoo et al. have indicated that there is a close relationship between gravity models and GNN [7]. Knowing this, Izumi et al. have proposed a method using GCN, and MLP, which will be discussed in the next section [1].

III. Baseline

A. Dataset

The current baseline for this problem was proposed by Izumi et al. [1]. They used the Centre d'Etudes Prospectives et d'Informations Internationales (CEPII) Gravity Database for their experiments [2]. The CEPII is a French research center specializing in studies of the world economy founded in 1978. This database holds a bounty of cultural, economic, and topographical information on all of the world's nations, as well as many variables with regards to their relationships such as distance, trade amounts, and political relations. This data is obtained through several reliable international organizations such as the World Bank, the World Trade Organization (WTO), and the International Monetary Fund (IMF) [1].

B. Architecture

Izumi et al. propose a model using GCN, a graph machine learning tool which embeds nodes by aggregating the information of itself and its neighbors, and feeding these embedded nodes into a decoder model to predict edge weights. This is opposed to many other experiments using GNNs that simply predict the existence of trade relations to a certain threshold. This method captures more complex relations between countries than shallow neural networks, and arguably makes more sense than previously used deep neural networks given the nature of the links between nations creating a graph structure. There are three main ideas behind their method.

First, they strongly rely on the gravity model to provide most of their data. In fact, the only features they included in their experiment are "exporter, importer, exporter-importer distance, exporter GDP, importer GDP, export values, and import values" [1]. This is because one of the goals of the experimenters was to test if machine learning tools could learn the relation stated in the gravity model.

They leverage this through a two layer GCN to aggregate the information of surrounding nodes, thus creating the set of embedded nodes. (2) shows how the output of each layer is calculated. In this equation, $\tilde{A}$ is the adjacency matrix with self-connections added, and $\tilde{D}$ is simply a degree matrix used to normalize $\tilde{A}$ in such a way that the nodes remain stable throughout the process. H_k is a vector of the embedded nodes at layer k of the process, with k=0 being the original nodes. $W^{(k)}$ is the trainable weight vector at layer k. To finalize the embedding process, the ReLU activation function is applied [1]. Izumi et al. used a two layer GCN for this experiment.

$$H_{k+1} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(k)}W^{(k)}) \tag{2}$$

Lastly they use a modified auto encoder, which they named the Gravity-informed Graph Auto-encoder to translate these weights into trade amount predictions [1]. While they hypothesized this approach would have the best results, their experiments found that feeding the resulting nodes of GCN concatenated with the edge data into an MLP resulted in the best outcome so this approach was selected as the baseline. This approach can be seen in (3), where $||$ represents the concatenation of vectors.

$$A^{amt}_{u,v} \approx MLP(H_u||H_v||Eu,v) \tag{3}$$

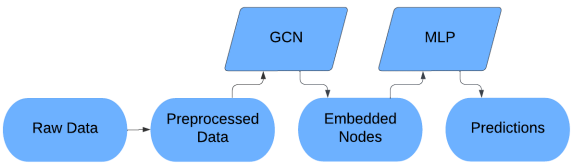

Fig. 1. Architecture diagram of the model proposed by Izumi et al.

To summarize, the model takes in the distances between each country, as well as the GDP of this country, and passes through a 2 layer GCN which encodes the data and outputs the embedded nodes. The MLP takes in the concatenation of the embedded nodes and the edge information as an input, and outputs the final predictions.

C. Evaluation

Izumi et al. evaluated their results using Root Mean Square Error (RMSE) which can be seen in (4). This is a simple measure of error which is commonly adopted for any model predicting a continuous output. Their experiments resulted in an RMSE of 4.122 on average.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}} \tag{4}$$

As the output in this model is in actuality the log of trade flow, this metric requires interpretation as an error in terms of the scale of the prediction as can be seen through the following calculations.

$$\begin{aligned} \epsilon &= ln(Predicted) - ln(Actual) \\ &= ln(\frac{Predicted}{Actual}) \\ e^{\epsilon} &= \frac{Predicted}{Actual} \end{aligned}$$

This shows that an error of ϵ signifies that the prediction is e^{ϵ} times the magnitude of the actual trade-flow.

D. Baseline Replication

The same CEPII dataset used by Izumi et al. was used to replicate the baseline for comparison [2]. It was unclear however what subset of the data they used. They note that their network contains 13811 edges, which likely means they pruned the dataset down to a single year. As a result, a single year (2018) was used in the following experiments, which resulted in a similarly sized graph. The extremely small RMSE they achieved seems unfeasible if they predicted trade amount in thousands of dollars (the unit used in the dataset). Therefore, it seems that they used a log transformation on the output, but it is unclear what log base they used. The replicated baseline described in the next paragraph achieved a significantly lower RMSE of approximately 2 which could be the result of a different log base, or a different architecture in the GCN or MLP stages of the model. While Izumi et al. used the Deep Graph Library (DGL) for the implementation of their GNNs, the StellarGraph library was used for the experiments in this paper, which could also lead to this difference [8]. It is also worth noting that the original baseline results are extremely close to the square of the replication results, which may mean that they in actuality reported MSE, rather than RMSE.

Izumi et al. did not report the exact architecture of their GCN or MLP, so some experiments were run to see what got the best results. Using a two layer GCN as was done in the baseline, the best results were found with an eight dimensional GCN embedding layer, although tweaking this made little difference. A single layer GCN was also experimented with as

there were signs that the GCN was perhaps over smoothing the nodes as will be discussed later, but found this also made little difference. For the MLP two hidden layers of size forty and twenty with relu activation functions were used. Before each convolutional layer, dropout layers are added with dropout equal to .5. The output layer takes the form of a single node as this is a regression task. The model was then applied to the data using mean squared error (MSE) as the loss function, and Adam as the optimizer.

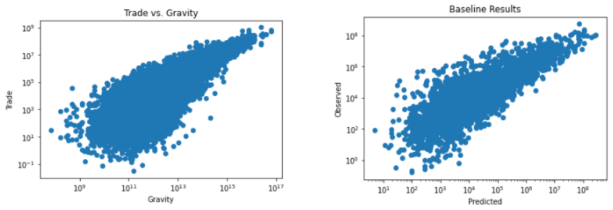

Fig. 2. The leftmost graph shows trade amounts plotted against the gravity function listed in (1) with the scalar constant equal to 1. The rightmost graph shows trade amounts plotted against the predictions of the replicated baseline model. It can be seen that the baseline somewhat models the gravity function. Both graphs are plotted in log-scale.

IV. Proposed Solution

Izumi et al. were able to predict trade amounts with higher accuracy than the previous base-line methods. While this method shows some promise, and confirms that GNNs can be used to model the behavior of international trade, and likely other gravity-like datasets, this observation seems fairly trivial as countries with higher GDP have more to trade, and trading with closer countries is inherently cheaper on average. It stands to reason that this approach would be far more useful if it included several other features of countries such as common languages, cultural information, and many other features of the countries. The first step in improving this baseline was to test how the current baseline performed if fed more variables. From there, it was decided to see how the MLP model would do without the GCN component, and met surprising results. This led to experimenting with a different approach replacing MLP with regression. Lastly, as the previous experiments seemed to imply GCN is not as effective as expected, it was replaced with a GraphSAGE model for embedding.

V. Experiments

A. Increased Predictors

First, this framework needed to be applied to more predictors to see how feasible it is for real world application. This came in the form of adding several new edge features. These are mostly binary variables indicating some relationship between the countries such as shared language, and shared religion. Others are indicators such as whether or not either country was ever a part of the other, or colonized by them. Interesting enough, using the same model parameters, the increased data led to an improvement in the training scores of the model, but resulted in worse results on the test set. This implies that something in the architecture led to over-fitting on the training set.

B. Simplified Model

To diagnose which part of the model was causing an over-fitting issue, the GCN component was cut from the model, leading to very surprising results. The new model at this point is simply a MLP on the raw data. Running this alternate model improved the results by extremely significant margins. In addition, testing this approach with only the gravity model variables had slightly better results than the GCN method on the same data.

Due to the nature of this dataset the graph is highly connected. Most nations are trading with each other to some extent due to the increased efficiency of transportation over the years, and the introduction of virtual goods. As a result, the GCN stage is likely over-smoothing the data, as most countries are now receiving information from every other nation, bringing every data point somewhat closer to each other, and taking away from what makes the node unique. This is a big issue, as the node information of the countries is one of the most important factors in determining trade amounts.

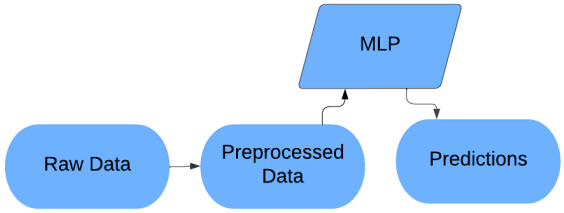

Fig. 3. Proposed Architecture for a simplified model

C. Linear Regression

After finding that more complex and demanding machine learning algorithms were achieving suspect results, it was decided to try a simpler method in the decoding stage of the algorithm. Linear regression was applied to the embedded nodes from GCN, and it was found that the results were the worst achieved by a significant degree. While this approach does not make sense to move forward with on this small dataset, it may be worth exploring for larger datasets, and perhaps after attempting some additional variable transformations.

The inferior accuracy of the linear regression model comes as no surprise. While MLP can learn many different types of complex relationships through many iterations and hidden layers, regression is limited to optimizing the scalar weights applied to each feature. (5) shows the final predictive model after regression is applied. In this equation, $\hat{Y}_i$ is the prediction, β_0 is the learned intercept, β is the learned scalar weights, X_i are the independent variables, and ϵ_i is an error term.

$$\hat{Y}_i = \beta_0 + \beta X_i + \epsilon_i \tag{5}$$

D. GraphSAGE

Finally, given that the GCN seemed to be leading to worse results for larger datasets, attention was shifted to replacing this with another encoder model. researching alternate encoding models led to GraphSAGE which provides several benefits. Most importantly, GraphSAGE is built to work on dynamic graphs and thus performs better than GCN when new information is added, or in this case applied to new data, without a full retraining of the model. This is due to the fact that GraphSAGE utilizes inductive learning, meaning that the model learns how to best embed nodes, and remembers this when fed new nodes. Another key difference is that GraphSAGE uses a generalized aggregation function whereas GCN uses a symmetric square normalization which closely approximates the mean aggregator. This aggregation method utilizes the mean aggregate in combination with several other aggregation techniques resulting in a more versatile aggregation. The GraphSAGE method not only achieves better results than the GCN method on this dataset, but there are several reasons to believe that it also makes the model more likely to be useful than GCN in many other applications, which will be discussed in the following sections.

Stellargraph was once again used for the implementation of GraphSAGE. The model was created with two hidden layers of size 8, similar to the GCN model, however no dropout was used in this case. The MLP component remained untouched for these experiments, to best capture the difference between GraphSAGE and GCN.

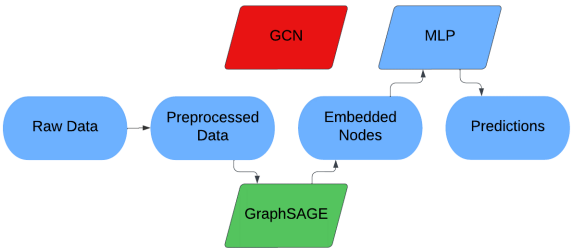

Fig. 4. Proposed architecture for a GraphSAGE reliant algorithm.

VI. Results

After completing all of the experiments, the results were quite surprising. It was found that the best performing model on the simple gravity data was the GraphSage model. Not only does the GraphSage lead to the best RMSE, it also is able to be used on dynamic datasets that are constantly changing. This is due to the fact that once the encoder is trained it can be used to encode new nodes without having to retrain on the whole graph unlike GCN. It can be seen however that the GraphSAGE model suffers from over-fitting even on the smaller dataset. This shows that the GraphSAGE implementation is a viable alternative to GCN, and is in fact an improvement over GCN in this case.

The most surprising thing however is that the simple MLP model performs the best on the dataset expanded to include additional features. It seems that the convolution leads to too much information loss, and that outweighs any benefit that sharing information between nodes may have had. Unfortunately, much like the GCN model, the GraphSAGE model cannot compete with the simple MLP model in terms of accuracy on this dataset.

	Input	Encoder	Decoder	RMSE		
				Train	Validation	Test
Baseline	Gravity features	GCN	MLP	1.998	1.986	2.03
	Increased features	GCN	MLP	1.779	2.011	2.095
Reduced	Gravity features	N/A	MLP	1.885	**1.904**	1.917
	Increased features	N/A	MLP	**1.39**	**1.33**	**1.379**
Regression	Gravity features	GCN	Linear Regression	2.414	2.348	2.361
GraphSAGE	Gravity features	GraphSAGE	MLP	**1.569**	1.952	**1.873**
	Increased features	GraphSAGE	MLP	1.618	1.902	1.894

Fig. 5. The results of the experiments. All evaluation metrics are RMSE described in (4).

VII. Conclusion

The goal of this project was to build upon the methods of Izumi et al. to make a better and more versatile model for gravity-like data, and international trade data in particular. Specifically, this project sought to improve upon the accuracy of international trade prediction, and then adapt it to be more useful for dynamic and larger datasets. Through the simplification of the model better results were achieved when working on the dataset with more features. This method however may not be ideal for larger problems with less connected graphs, as the convolution will likely be more useful in this case. This was addressed by replacing the GCN component of the baseline with the GraphSage method, leading to better results. While it did not perform as well as the simplified model on the larger data set, it did perform better on the small dataset. Therefore, there is reason to believe it may be more useful on a dataset with more data points and more data stored in the nodes rather than the edges, as GraphSage currently does not utilize edge features.

Moving forward, re-purposing GraphSage to leverage edge features could be explored, as GraphSage shows a lot of promise with this dataset, but is unable to utilize much of the international data which is stored in the edge. After evaluating this, testing could be done to see how the model generalizes to other gravity-like datasets to see how it performs at a larger scale. Using GraphSage could allow the model to be useful in many other more dynamic problems such as traffic control and the stock market, as these are constantly evolving, making GCN unfeasible in these applications. Alternatively to adapting GraphSAGE to better utilize edge features, other convolutional methods could be explored. Cui et al. proposed an edge-enhanced GCN which looks promising in this implementation [9].

REFERENCES

[1] K. Izumi, N. Minakawa, and H. Sakaju. "Bilateral Trade Flow Prediction by Gravity-informed Graph Auto-encoder," in 2022 IEEE International Conference on Big Data, 2022, pp. 2327-2332. [Online].

[2] Bouët, Antoine. "CEPPI Research and Expertise on the World Economy." CEPII.

[3] I. Wohl and J. Kennedy, "Neural network analysis of international trade," Office of Industries Working Paper ID-049, May 2018

[4] E. M. Kottou, T. A. Grubelich, and X. Wang, "Bilateral trade flow prediction models enhanced by wavelet and machine learning algorithms," in 2020 International Conference on Computational Science and Computational Intelligence (CSCI), 2020, pp. 1510–1516.

[5] D. Karemera, V. I. Oguledo, and B. Davis, "A gravity model analysis of international migration to north america," Applied Economics, vol. 32, no. 13, pp. 1745–1755, 2000.

[6] M. A. Sayed, M. M. Rahman, M. I. Zaber, and A. A. Ali, "Understanding dhaka city traffic intensity and traffic expansion using gravity model," in 2017 20th International Conference of Computer and Information. Technology (ICCIT), 2017, pp. 1–6.

[7] K. Panford-Quainoo, A. J. Bose, and M. Defferrard, "Bilateral trade modelling with graph neural networks," in Practical ML for Developing Countries Workshop, ser. ICLR '20, 2020.

[8] C. Data61, 'StellarGraph Machine Learning Library', GitHub Repository. GitHub, 2018.

[9] S. Cui et al., "Edge-Enhanced Graph Convolution Networks for Event Detection with Syntactic Relation," Association for Computational Linguistics 2020.