Shortest Path Distance Approximation Using Deep Learning Techniques

3 authors, including:

Fatemeh Salehi Rizi
Universität Passau
19 PUBLICATIONS   **89** CITATIONS

Michael Granitzer
Universität Passau
298 PUBLICATIONS   **2,300** CITATIONS

Some of the authors of this publication are also working on these related projects:

Project   A Fast Survey on Methods for Classification Anonymity Requirements View project

Project   provenance analytic View project

Shortest Path Distance Approximation using Deep learning Techniques

Fatemeh Salehi Rizi
Department of Computer Science and Mathematics
University of Passau
Passau, Germany
Fatemeh.SalehiRizi@uni-passau.de

Joerg Schloetterer
Department of Computer Science and Mathematics
University of Passau
Passau, Germany
Joerg.Schloetterer@uni-passau.de

Michael Granitzer
Department of Computer Science and Mathematics
University of Passau
Passau, Germany
Michael.Granitzer@uni-passau.de

Abstract—Computing shortest path distances between nodes lies at the heart of many graph algorithms and applications. Traditional exact methods such as breadth-first-search (BFS) do not scale up to contemporary, rapidly evolving today's massive networks. Therefore, it is required to find approximation methods to enable scalable graph processing with a significant speedup. In this paper, we utilize vector embeddings learnt by deep learning techniques to approximate the shortest paths distances in large graphs. We show that a feedforward neural network fed with embeddings can approximate distances with relatively low distortion error. The suggested method is evaluated on the Facebook, BlogCatalog, Youtube and Flickr social networks.

Index Terms—Shortest Path Distance, Deep Learning, Graph Embedding

I. Introduction

Finding shortest path distances between nodes in a graph is an important primitive in a variety of applications. For instance, the number of links between two URLs indicates page similarity in a graph of the Web [1]. In a semantic web ontology, shortest path distances among entities are used for ranking their relationships [2]. The number of hops from one person to another indicates the level of trust in a trust network [3]. In social networks, the shortest path distance is used to compute the closeness centrality [4].

Traditional methods for computing node distance do not scale with graph size. For a graph with n nodes and m edges, efficient implementations of Dijkstra compute the shortest paths for a node to others in $O(n \log n + m)$ time. A slight generalization of Dijkstra, known as the A^* algorithm, uses heuristic techniques for computing shortest distances. In practice, A^* works at least as quickly as Dijkstra's algorithm, however, the run time complexity is still $O(n \log n + m)$. Tolerable for small graphs, but on a large million node graph computation can take up to a minute for a single node distance [5]. Given the high cost of storing precomputed distances, researchers have limited choice but to sample subgraphs or seek approximate results. For many practical applications, finding out approximate distances between nodes can be sufficient.

In this paper, we propose a novel method for approximating shortest path distance measurements between two nodes utilizing vector embeddings generated by deep learning techniques.

Node2vec and Poincarè embeddings are recently studied for the basic graph analysis tasks such as link prediction [6], [7] and node classification [6], [8]. This is due to the fact that they speed up the computation time and yield precise results compared to the traditional matrix factorization techniques. We thus exploit these two embeddings to approximate shortest path distances. At a high level, we first learn embeddings using node2vec [9] and Poincaré [10] for every single node in the graph. We then follow the "landmark-based" approach proposed in [11], where we choose a small number of nodes as landmarks. We compute the actual shortest paths distances from each landmark to all of the remaining nodes. We use these pairs as our training set. Finally, we train a feedforward neural network with embedding vectors to approximate the distance between corresponding nodes. Feedforward networks are well known for their good representational capability [12] and should be sufficient to approximate shortest path distance functions giving graph embeddings.

Due to the sparsity property of real-world (social) networks, the number of edges is proportional to the number of nodes. Therefore, generating the ground truth using the landmark-based approach takes a linear time complexity which yields a linear running time for the entire proposed method. More details are elaborated in section III.

Overall, we study advanced graph embedding techniques to make the following contributions:

- We theoretically motivate and suggest the use of node2vec and Poincaré embeddings for node distance approximation in large graphs (section III).
- We show that neural networks can predict the shortest path distances effectively and efficiently, especially for shorter paths (section IV).
- We demonstrate that different embedding techniques as well as different parameter settings have a significant influence on the approximation quality (section IV).
- We compare our approximations to the most prominent works in the state-of-the-art and show, that our approach outperforms current methods in terms of prediction accuracy (section IV).

In order to demonstrate the performance of our method

in practice, we conduct experiments with four real-world datasets. The experiments indicate, that our predictor performs better for shorter paths. However, embeddings behave poorly to approximate the longer paths. The reason lies in both the embedding technique and the sampling strategy for training pairs. We provide more details in section IV.

The rest of the paper is structured as follows. Section II provides an overview of recent existing shortest path distance approximation and graph embedding techniques. Our proposed method is elaborated in section III. Section IV summarizes the experimental evaluation results. Finally, section V concluding remarks.

II. BACKGROUND AND RELATED WORK

A. Shortest Path

The task of computing the shortest path distance from a single node to all other nodes is known as *single source shortest paths (SSSP)*. Exact methods such as Dijkstra compute SSSP for weighted graphs with n nodes and m edges in time $O(m + n \log n)$. For unweighted sparse graphs, shortest paths can be computed using Breadth First Search (BFS) in time $O(m + n)$. However, exact methods, are extremely slow for performing queries on today's very large online networks. For many practical applications, finding out approximate distances between nodes can be sufficient. Among the approximate methods, a family of scalable algorithms for this problem are the so-called landmark-based approaches [11]. In this family of techniques, a fixed set of landmark nodes is selected and actual shortest paths are precomputed from each landmark to all other nodes. Knowledge of the distances to the landmarks, together with the triangle inequality, typically allows one to compute approximate distance between any two nodes in $O(l)$ time, where l is the number of landmarks. [11]. Although landmark-based algorithms do not provide strong theoretical guarantees on approximation quality [13], they have been shown to perform well in practice, scaling up to graphs with millions of edges with acceptable accuracy and response times of under one second per query [14].

Inspired by landmark-based methods, authors in [11] and [15], suggested the idea of graph coordinate systems, which embeds graph nodes into points on a coordinate space. The resulting coordinates can be used to quickly approximate node distance queries on the original graph. However, it has several limitations in practice. First, their initial graph embedding process is centralized and computationally expensive, which presents a significant performance bottleneck for larger graphs. Second, their results produce relatively high error rates, which limits the types of applications it can serve. Finally, it is unable to produce actual paths connecting node pairs, which is often necessary for a number of graph applications.

B. Graph Embedding

Besides the aforementioned methods, embedding the nodes with the explicit objective of preserving the shortest path length, various methods to embed the graph in general (or dimension reduction respectively) have been proposed (c.f.

Goyal and Ferrara [16] for a survey). Among the classical methods are Principal Component Analysis (PCA) [17], Linear Discriminant Analysis (LDA) [18], ISOMAP [19], Multi-dimensional Scaling (MDS) [20], LLE [21] and Laplacian Eigenmap [22] (c.f. Yan et al. [23] for a survey). However, most of these methods typically rely on solving eigen decomposition and the complexity is at least quadratic in the number of nodes, which makes them inefficient to handle large-scale networks.

Recently, neural network based approaches have been proposed, which were inspired by the ideas of Word2Vec [24], which is build around the distributional hypothesis, stating that words in similar contexts tend to have similar meaning [25]. DeepWalk [26] samples random walks from the graph and treats them as sentence equivalents. That is, given the representation of a node in the embedding space, DeepWalk approximates the conditional probability of nodes in the the the neighborhood. Thereby, nodes sharing a similar neighborhood, tend to have a similar representation in the embedding space. Similar to Word2Vec implicitly factorizing a matrix of word co-occurrences [27], [28], Deepwalk has been shown to factorize a matrix of node transition probabilities [29]. Node2vec [6] extends Deepwalk by introducing parameters to control the random walk behaviour. At the most extreme parameter choices, node2vec employs breadth-first or depth-first sampling, exploring the close-by neighborhood or nodes that are far apart in the network. LINE [30] explicitly optimizes the embeddings to capture first- and second-order proximity, by training separate embeddings for them, which are finally concatenated. First-order proximity is given by explicit connections between nodes, while second-order proximity is given by comparing the nodes' neighborhoods. Nickel and Kiela [7] embed the graph into a hyperbolic space, or more precisely into an n-dimensional Poincaré ball, capturing hierarchy and similarity. Several approaches have been proposed to better model long distance relationships or higher order proximity respectively. Instead of approximating the k-order proximity matrix, as DeepWalk does, GraRep [31] calculates it accurately, at the cost of increased complexity. Yang et al. [32] alleviate this problem by using information from lower order proximity matrices. The authors of HOPE [33] experimented with different similarity measures, such as Katz Index, Rooted Page Rank and Adamic-Adar. HARP [34] and Walklets [35] address capturing higher-order proximity by adapting the random walk strategy. While HARP coarsens the graph and learns representations via hierarchically collapsed graphs, Walklets skips over steps in the random walks. Deep architectures have been proposed, aiming at capturing non-linearity in the graphs. SDNE [36] and DNGR [37] utilize autoencoders, GCN [38] defines a convolution operator on the graph.

To our best knowledge, approximating the shortest path distances based on embeddings that have not been specifically tailored towards this end, has not been investigated yet.

III. APPROACH

A. Distance Approximation

Let $G = (V, E)$ be an unweighted undirected graph with n nodes and m edges. Graph embedding techniques create a real-valued, the so called vector embedding $\phi(v) \in R^d$ for every node $v \in V$. Given a pair of nodes $u, v \in V$ with the real shortest path distance $d_{u,v}$, the goal is to approximate the distance as $\hat{d}$ using a feedforward neural network. Formally, we define $\hat{d}$ as function

$$\hat{d} : \phi(u) \times \phi(v) \mapsto R^+$$

that maps a pair of vector embeddings to a real-valued shortest path distance $d_{u,v}$ in G.

To train the neural network, we need to extract training pairs from the entire graph G. We first choose a small number of l nodes as landmarks, where $l \ll n$. We then compute the actual shortest distances from each landmark to all of the remaining nodes using BFS. It yields $l(n - l)$ training pairs. Given a training pair $< \phi(v), \phi(u) >$, we create a joint representation as input to the neural network by applying a binary operation, namely subtraction, concatenation, average and point-wise multiplication, over the vector embeddings. The definitions of the binary operations are listed in the Table I. Eventually, vectors of the training set serve as input for a feedforward neural network. The neural network maps the input vectors to a real-valued distance.

TABLE I: Choice of binary operators correspond to the ith component of ϕ

Operator	Symbol	Definition
Subtraction	$\ominus$	$\phi_i(u) - \phi_i(v)$
Concatenation	$\oplus$	$(\phi(u), \phi(v))$
Average	$\oslash$	$\frac{\phi_i(u) + \phi_i(v)}{2}$
Hadamard	$\odot$	$\phi_i(u) * \phi_i(v)$

Our feedforward network consists of an input layer, a hidden layer, and an output layer. The size of the input layer depends on the binary operation on vector embeddings. For example subtraction needs d neurons while concatenation requires $2d$. We set the rectified unit (ReLU) [39] as activation function for the first two layers. ReLU does not face gradient vanishing problem and it has been shown that deep networks are trained efficiently using ReLU. Since the network does a regression task, the output layer is a single unit of softplus [40] which is a smoother version of ReLU with the range of $[0, \infty]$. We assess the quality of predictor by Mean Squared Error (MSE) which measures the average of the squares of difference between the estimator and what is estimated. As optimizer, we use Stochastic Gradient Descent (SGD) [41] which is usually fast and efficient for large-scale learning.

B. Computational Complexity

The proposed method achieves a linear runtime complexity. First, we learn vector embeddings which takes precomputation time $O(n)$ where n is the number of nodes in the graph [16].

We then use the landmark-based scheme to minimize the number of shortest path computations needed to establish the ground truth. By choosing a small, constant number of landmarks, we only need to compute a BFS tree for each landmark. The resulting values represent shortest distances from all remaining nodes to these landmarks, and are sufficient to compose the training set. With l nodes as landmarks, where $l \ll n$, we have $l(n - l)$ training pairs. It takes time $O(l(n + m))$, knowing BFS on unweighted sparse graphs consumes $O(n + m)$ time. The advantage of using a graph embedding is that a feedforward neural network can answer a distance query between two nodes u, v in a small amount of time independent of the graph size, i.e. $O(1)$ time. So calculating the shortest path distances from a starting node u to all other nodes takes $O(n)$.

IV. EXPERIMENTAL EVALUATION

In this section, we report the performance of the proposed method. We evaluate node2vec and Poincaré embeddings for distance approximation applying the four different binary operations. We start by describing our datasets, and then we set the hyperparameters. Eventually, we describe our verification approach along with discussions over the results.

A. Datasets

We test our method on four real-world social network graphs, representing four different orders of magnitude in terms of network size.

- **Facebook.** This dataset consists of friends lists from Facebook. Facebook data was collected from survey participants using a Facebook app. Facebook data has been anonymized by replacing the Facebook internal ids for each user with a new value [42].
- **BlogCatalog.** This is a network of social relationships of the bloggers listed on the BlogCatalog website. The labels represent blogger interests inferred through the metadata provided by the bloggers [43].
- **Youtube.** This is the friendship network of the video-sharing site Youtube. Nodes are users and an undirected edge between two nodes indicates a friendship [44].
- **Flickr.** This dataset is built by forming links between images sharing common metadata from Flickr. Edges are formed between images from the same location, submitted to the same gallery, group, or set, images sharing common tags, images taken by friends [45].

The properties of the datasets are summarized in Table II. The table shows the number of nodes n, number of edges m and average shortest distance between nodes $\bar{d}$. The most relevant previous works using a similar methodology, Orion [11] and Rigel [15], used Flickr as a common dataset in their evaluations. We therefore compare our results with the state-of-the-art on the Flickr dataset.

B. Parameters and Environment

To generate node2vec embeddings, we use the default settings [6] with the number of walks $\gamma = 10$ and length

TABLE II: Statistics of Social Network Datasets

Dataset	Nodes	Edges	$\bar{d}$
Facebook	4,039	88,234	4.31
BlogCatalog	10,312	333,983	2.72
Youtube	1,134,890	2,987,624	5.55
Flickr	1,715,255	15,551,250	5.13

$l = 80$. We tuned the context size setting it finally to $k = 5$ for learning node representations by predicting more nearby nodes. We also set $p = q = 1$ to explore local and global nodes equally. For Poincaré, we follow the default settings as $r > 1$, $t = 0.01$ [7], and 50 iterations.

We consider two different embedding dimensions $d = 32$ and $d = 128$ to investigate the impact of more features on distance approximation. To train the neural network, we initialize weights randomly and set the learning rate $lr = 0.01$ running 15 iterations. We run experiments on Linux machines with five 3.50GHz Intel Xeon(R) CPUs and 16GB memory.

C. Approximation Quality

Since shortest path distances are discrete real values, we first round the results of the prediction. We then measure the accuracy of our method using two key metrics. The first is the Mean of Relative Error (MRE). The Relative Error is widely used in the study of shortest path evaluation and defined as,

$$RE = \frac{|\hat{d} - d|}{d}$$

where d is the actual distance measured by BFS algorithm and $\hat{d}$ the approximation [11], [13], [14]. The MRE has a natural tendency to be smaller for larger distance values, which is not a perfectly fair comparison. Therefore, as a second evaluation measure we use the Mean Absolute Error (MAE) which is the most natural measure of average error magnitude independent of the target value.

D. Training and Test Data

In order to capture training pairs, we require computing BFS trees rooted from each landmark to all other remaining nodes. For the smaller datasets such as Facebook and BlogCatalog, we set $l = 100$ while for the two others 5 landmarks are enough. We omit paths with length 1 since the time complexity of finding direct neighbors is already linear $O(n)$ [46].

To avoid an imbalanced training set and perform a fast sampling, we downsample the minority classes (i.e. classes with few samples). In detail, when we search the shortest path for a given training pair by BFS, we also retrieve the nodes are included in this path. Respecting the fact that a shortest path carries nodes by keeping shortest distance between them. Additionally, in complex networks such as social networks the average distance is very small therefore the distribution of long paths is quite low [47]. In Figure 1, we plot histograms that reflect the distribution of training pairs in all four datasets. In BlogCatalog the average shortest path distance is 2.72 and

the distribution of path longer than 5 is prohibitively low. We therefore limit the path length to 5.

To compose the training matrix, we need to perform binary operations on the embedding vectors of given training pairs. We do Subtraction, Concatenation, Average and Hadamard on pairs of vectors.

For test pairs, we do the same strategy as training pairs considering a smaller set of landmarks. We start BFS traversals from landmarks to other remaining nodes which generate a set of unseen pairs. Overall, we gather around $100,000$ unseen test pairs for each dataset. The statistics of training and test pairs are available in Table III.

TABLE III: Statistics of Training and Test data

Dataset	Nodes	Training pairs	Test pairs
Facebook	4,039	1,022,640	109,978
BlogCatalog	10,312	1,409,700	88,316
Youtube	1,134,890	2,452,757	184,413
Flickr	1,715,255	2,579,437	112,967

E. Baseline

As a baseline, we conduct an experiment where the predictor is a simple linear regression. The input of the predictor is embeddings learnt by node2vec or Poincaré and the output is distance values between pair of nodes. Since the state-of-the-art [16] observes the best performance with embedding dimensions $d = 128$ in primitive graph analysis tasks (e.g. link prediction, node classification), we set $d = 128$ in our baseline experiment. However, we study the effect of dimension while we predict the distance using neural networks. Table IV shows MAE values for distance prediction besides the impact of different binary operations. We observe that node2vec achieves lower error values than Poincaré specifically when we take the average of embedding vectors as the input of the predictor.

TABLE IV: Mean Absolute Error (MAE) of shortest paths return by a Linear Regression

Dataset	Embedding	MAE			
		$\ominus$	$\oplus$	$\oslash$	$\odot$
Facebook	node2vec	0.679	0.663	**0.546**	0.653
	Poincaré	0.801	0.788	0.656	0.767
BlogCatalog	node2vec	0.483	0.453	**0.407**	0.423
	Poincaré	0.417	0.450	0.436	0.447
Youtube	node2vec	0.708	0.722	**0.695**	0.739
	Poincaré	0.983	1.267	1.195	1.099
Flickr	node2vec	0.633	0.694	**0.574**	0.739
	Poincaré	0.944	0.891	0.831	0.822

F. Results and Discussion

Using a feedforward neural network, we calculate the mean relative and the mean absolute error during the test phase.
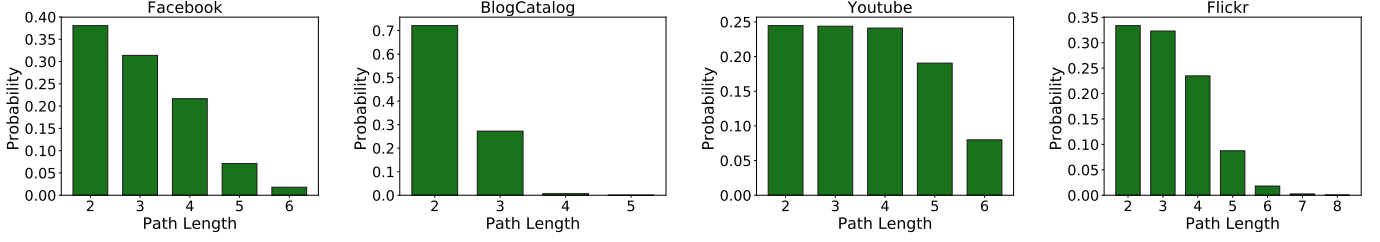
Fig. 1: Distributions of distances in the training set with downsampling for all datasets

The obtained error values for different embedding techniques, different embedding sizes and the different binary operators are provided in Table V. The best results are achieved by node2vec embeddings of size 128 with an average MAE of 15% across all datasets. The results indicate, that independent of the network size, our method achieves a quite low approximation error compared to the baseline.

a) Embedding Techniques and Embedding Size: The number of dimensions play an important role. Although higher dimensions produce smaller errors, as the dimension increases the time for generating embeddings and distance approximation increases as well.

Reducing the size of embeddings in hierarchically organized networks was one goal underlying the development of Poincarè embeddings. Poincarè use a Hyperbolic space to alleviate overfitting and complexity issues that Euclidean embeddings face especially if the data has intrinsic hierarchical structure. Scale-free graphs in general and Social Networks in particular have been known to form innate hierarchical structures [48], which motivates the use of Poincarè embeddings. Moreover, we would expect Poincarè embeddings to more accurately represent distances than node2vec embeddings in Euclidean space. However, Table V shows that the error observed for node2vec embedding is remarkably lower than Poincarè embedding.

The features learned in node2vec are fundamentally tied to the random walk strategy. Node2vec adopts short random walks to explore therefore it learns the structure of local neighborhoods. In scale-free networks, the average shortest path distance grows logarithmically [49], hence the shortest path distance stays as a local feature. According to the result in Table V, node2vec successfully learns distances specifically for embedding size 128.

While, Poincarè makes use of hyperbolic spaces to encode both hierarchy and semantic similarity into a Poincarè ball. Regarding our results, Poincarè cannot capture features related to distances of nodes belong to different hierarchies. Table V demonstrates that increasing dimensionality in Poincarè ball do not effect the accuracy of prediction in all datasets.

b) Error Distribution over Path Lengths: We explore the accuracy of predictions for paths of different lengths. Figure 2 shows the mean absolute errors per path length on three graphs. Error values are estimated for different binary operations on embeddings of size 128. Observe that the larger

errors caused by longer paths utilizing node2vec embeddings. In one hand, we do not have enough samples for longer distances in the training set. On the other hand, node2vec fails to learn structural features of faraway nodes. Similarly, long paths cause high errors using Poincarè embedings in all three datasets. It can be the effect of locality property of the Poincarè distance which places leaf nodes to the boundary. Therefore, the distance between leaf nodes of different hierarchies cannot be preserved according the original graph neighborhood.

c) Effect of Binary Operators: To generate feature representations of training pairs, we compose the learned embeddings of the individual nodes using simple binary operators. The binary operators are listed in Table I. This compositionality lends node2vec and Poincaré to the prediction task involving nodes. As Table V shows, binary operators do not have a consistent behavior over different datasets and different dimension sizes. For instance, the average operator outperforms others in Facebook graph while concatenation works better for Youtube dataset. As a future work, we would like to explore the reasons behind the unstable behavior of operators as well as effect of some other operators on prediction.

d) Comparison to the State-of-the-Art: In Figure 3, we plot the MAE for different path lengths using our method against the two state-of-the-art methods Rigel [15] and Orion [11] on the Flickr dataset. Errors are calculated when the input of the predictor is node2vec embeddings of size 128. We can observe that Orion shows the highest MAE value for all paths. In general, our method consistently and significantly outperforms Rigel specifically for shorter paths.

V. CONCLUSION AND FUTURE WORK

Traditional methods for computing shortest path distances no longer scale to today's massive graphs with millions of nodes and billions of edges. Motivated by landmarked-based approaches, we propose a new method that approximates node distances by first embedding graphs into an embedding space. We utilized two recent graph embedding techniques and fed their vectors into a feedforward neural network. The results are impressive. Our method produces shortest distances for the large majority of node pairs, matching the most accurate of ground truth. And it does this quickly, returning results in a linear time.

We plan to handle longer paths in larger graphs, where we face the higher errors. One way is to apply other embedding

TABLE V: Mean Absolute Error (MAE) and Mean Relative Error (MRE) of shortest paths utilizing different embedding techniques

| Dataset	Embedding	Size	MAE				MRE			
			⊖	⊕	⊘	⊙	⊖	⊕	⊘	⊙
Facebook	node2vec	32	0.480	0.415	0.233	0.531	0.175	0.164	0.068	0.188
		128	0.197	0.258	**0.118**	0.217	0.071	0.099	**0.038**	0.081
	Poincaré	32	0.592	0.594	0.552	0.604	0.214	0.211	0.218	0.212
		128	0.437	0.315	0.372	0.608	0.169	0.115	0.142	0.246
BlogCatalog	node2vec	32	0.277	0.242	0.197	0.193	0.092	0.103	0.067	0.067
		128	0.220	0.275	0.159	**0.154**	0.077	0.119	0.064	**0.059**
	Poincaré	32	0.338	0.338	0.343	0.338	0.108	0.108	0.112	0.108
		128	0.331	0.354	0.277	0.338	0.115	0.138	0.097	0.108
Youtube	node2vec	32	0.676	0.265	0.455	0.625	0.230	0.066	0.163	0.223
		128	0.344	**0.154**	0.174	0.244	0.101	**0.034**	0.040	0.061
	Poincaré	32	1.095	0.708	1.134	0.774	0.429	0.264	0.446	0.291
		128	1.270	1.185	1.746	0.771	0.497	0.468	0.681	0.262
Flickr	node2vec	32	0.699	0.295	0.564	0.525	0.250	0.086	0.183	0.198
		128	0.238	**0.168**	0.181	0.222	0.171	**0.074**	0.178	0.179
	Poincaré	32	0.995	0.808	1.022	0.874	0.349	0.284	0.429	0.278
		128	0.803	0.662	0.807	0.764	0.397	0.432	0.566	0.364
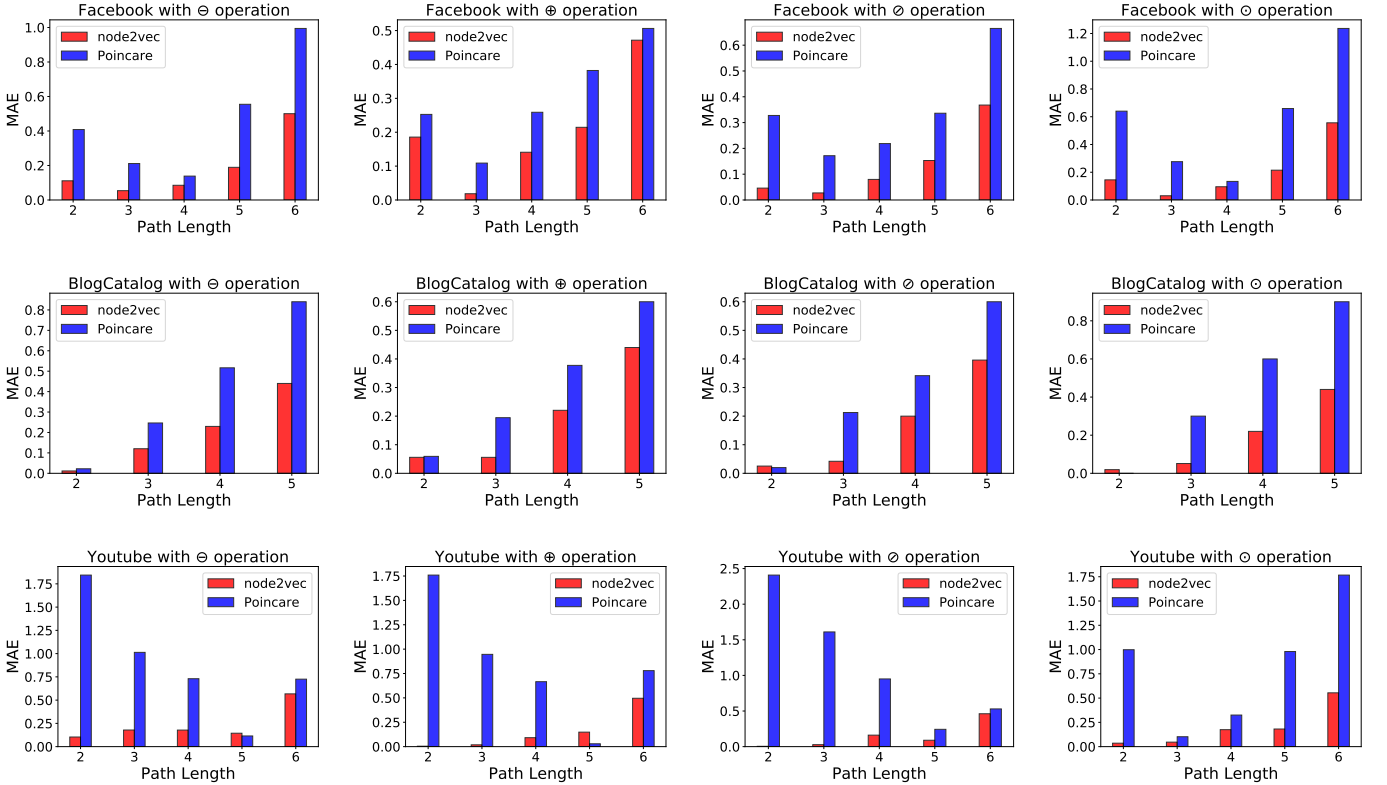
Fig. 2: Mean Absolute Error (MAE) of different path lengths comparing node2vec and Poincaré embeddings with size 128 for Facebook, BlogCatalog and Youtube datasets.
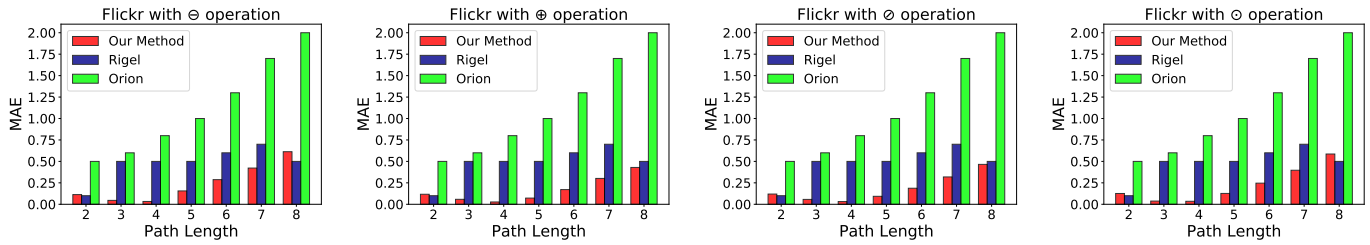
Fig. 3: Mean Absolute Error (MAE) of shortest paths returned by our method, Orion and Rigel on the Flickr dataset.

techniques such as HARP [34] which learns the structure of neighborhoods using graph coarsening. Another way is to reach a balance training set and to use other neural networks such as Siamese neural networks [50].

ACKNOWLEDGMENT

REFERENCES

[1] A. Das Sarma, S. Gollapudi, M. Najork, and R. Panigrahy, "A sketch-based distance oracle for web-scale graphs," in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 401–410.

[2] K. Anyanwu and A. Sheth, "P-queries: enabling querying for semantic associations on the semantic web," in *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003, pp. 690–699.

[3] G. Swamynathan, C. Wilson, B. Boe, K. Almeroth, and B. Y. Zhao, "Do social networks improve e-commerce?: a study on social marketplaces," in *Proceedings of the first workshop on Online social networks*. ACM, 2008, pp. 1–6.

[4] M. V. Vieira, B. M. Fonseca, R. Damazio, P. B. Golgher, D. d. C. Reis, and B. Ribeiro-Neto, "Efficient search ranking in social networks," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. ACM, 2007, pp. 563–572.

[5] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis, "Fast shortest path distance estimation in large networks," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 867–876.

[6] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 855–864.

[7] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 6341–6350. [Online]. Available: http://papers.nips.cc/paper/7213-poincare-embeddings-for-learning-hierarchical-representations.pdf

[8] F. S. Rizi, M. Granitzer, and K. Ziegler, "Global and local feature learning for ego-network analysis," in *Proceedings of the 28th International Workshop on Database and Expert Systems Applications (DEXA)*. IEEE, 28-31 Aug. 2017.

[9] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[10] M. Nickel and D. Kiela, "Poincar\'e embeddings for learning hierarchical representations," *arXiv preprint arXiv:1705.08039*, 2017.

[11] X. Zhao, A. Sala, C. Wilson, H. Zheng, and B. Y. Zhao, "Orion: shortest path estimation for large social graphs," *networks*, vol. 1, p. 5, 2010.

[12] J. Bilski, "The ud rls algorithm for training feedforward neural networks," *International Journal of Applied Mathematics and Computer Science*, vol. 15, pp. 115–123, 2005.

[13] J. Kleinberg, A. Slivkins, and T. Wexler, "Triangulation and embedding using small sets of beacons," in *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*. IEEE, 2004, pp. 444–453.

[14] A. Gubichev, S. Bedathur, S. Seufert, and G. Weikum, "Fast and accurate estimation of shortest paths in large graphs," in *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 2010, pp. 499–508.

[15] X. Zhao, A. Sala, H. Zheng, and B. Y. Zhao, "Efficient shortest paths on massive social graphs," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on*. IEEE, 2011, pp. 77–86.

[16] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *arXiv preprint arXiv:1705.02801*, 2017.

[17] I. T. Jolliffe, "Principal component analysis and factor analysis," in *Principal component analysis*. Springer, 1986, pp. 115–128.

[18] A. M. Martínez and A. C. Kak, "Pca versus lda," *IEEE transactions on pattern analysis and machine intelligence*, vol. 23, no. 2, pp. 228–233, 2001.

[19] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[20] J. B. Kruskal and M. Wish, *Multidimensional scaling*. Sage, 1978, vol. 11.

[21] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[22] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, 2002, pp. 585–591.

[23] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 1, pp. 40–51, 2007.

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[25] Z. Harris, "Distributional structure," *Word*, vol. 10, no. 23, pp. 146–162, 1954.

[26] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[27] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[28] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, pp. 2177–2185.

[29] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, pp. 2111–2117.

[30] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the*

24th International Conference on World Wide Web, ser. WWW '15. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[31] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 891–900.

[32] C. Yang, M. Sun, Z. Liu, and C. Tu, "Fast network embedding enhancement via high order proximity approximation," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, 2017, pp. 19–25.

[33] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding." in *KDD*, 2016, pp. 1105–1114.

[34] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, "Harp: Hierarchical representation learning for networks," in *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

[35] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, "Don't walk, skip!: Online learning of multi-scale network embeddings," in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. ACM, 2017, pp. 258–265.

[36] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1225–1234.

[37] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations." in *AAAI*, 2016, pp. 1145–1152.

[38] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations (ICLR-17)*, 2017.

[39] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[40] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[41] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[42] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Advances in neural information processing systems*, 2012, pp. 539–547.

[43] R. Zafarani and H. Liu, "Social computing data repository at ASU," 2009. [Online]. Available: http://socialcomputing.asu.edu

[44] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.

[45] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 2007, pp. 29–42.

[46] U. Kang, S. Papadimitriou, J. Sun, and H. Tong, "Centralities in large networks: Algorithms and observations," in *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM, 2011, pp. 119–130.

[47] T. Akiba, Y. Iwata, and Y. Yoshida, "Fast exact shortest-path distance queries on large networks by pruned landmark labeling," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 2013, pp. 349–360.

[48] E. Ravasz and A.-L. Barabási, "Hierarchical organization in complex networks," *Physical Review E*, vol. 67, no. 2, p. 026112, 2003.

[49] A.-L. Barabási, *Network science*. Cambridge university press, 2016.

[50] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, vol. 2, 2015.